Communicating Piecewise Deterministic Markov Processes

Strubbe, Stefan; Schaft, Arjan van der

Published in:
Stochastic Hybrid Systems: Theory and Safety Critical Applications

Publication date:
2006

Communicating Piecewise Deterministic Markov Processes

Stefan Strubbe[1] and Arjan van der Schaft[1]

Department of Applied Mathematics, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands,
s.n.strubbe@math.utwente.nl, a.j.vanderschaft@math.utwente.nl

Summary. In this chapter we introduce the automata framework CPDP, which stands for Communicating Piecewise Deterministic Markov Processes. CPDP is developed for compositional modelling and analysis for a class of stochastic hybrid systems. We define a parallel composition operator, denoted as $|_A^P|$, for CPDPs, which can be used to interconnect component-CPDPs, to form the composite system (which consists of all components, interacting with each other). We show that the result of composing CPDPs with $|_A^P|$ is again a CPDP (i.e., the class of CPDPs is closed under $|_A^P|$). Under certain conditions, the evolution of the state of a CPDP can be modelled as a stochastic process. We show that for these CPDPs, this stochastic process can always be modelled as a PDP (Piecewise Deterministic Markov Process) and we present an algorithm that finds the corresponding PDP of a CPDP. After that, we present an extended CPDP framework called value-passing CPDP. This framework provides richer interaction possibilities, where components can communicate information about their continuous states to each other. We give an Air Traffic Management example, modelled as a value-passing CPDP and we show that according to the algorithm, this CPDP behavior can be modelled as a PDP. Finally, we define bisimulation relations for CPDPs. We prove that bisimilar CPDPs exhibit equal stochastic behavior. Bisimulation can be used as a state reduction technique by substituting a CPDP (or a CPDP component) by a bisimulation-equivalent CPDP (or CPDP component) with a smaller state space. This can be done because we know that such a substitution will not change the stochastic behavior.

1 Introduction

Many real-life systems nowadays are complex hybrid systems. They consist of multiple components 'running' simultaneously, having both continuous and discrete dynamics and interacting with each other. Also, many of these systems have a stochastic nature. An interesting class of stochastic hybrid systems is formed by the Piecewise Deterministic Markov Processes (PDPs), which were introduced in 1984 by Davis (see [3, 4]). Motivation for considering PDP systems is two-fold. First, almost all stochastic hybrid processes

H.A.P. Blom, J. Lygeros (Eds.): Stochastic Hybrid Systems, LNCIS 337, pp. 65–104, 2006.

that do not include diffusions can be modelled as a PDP, and second, PDP processes have nice properties (such as the strong Markov property) when it comes to stochastic analysis. (In [4] powerful analysis techniques for PDPs have been developed). However, PDPs cannot communicate or interact with other PDPs. In order to let PDPs communicate and interact with other PDP's the aim of this paper is to develop a way of opening the structure of PDPs accordingly to this purpose.

In this chapter we present a theory of the automata framework Communicating Piecewise Deterministic Markov Processes (CPDPs, introduced in [12]). A CPDP automaton can be seen as a PDP type process enhanced with interaction/communication possibilities (see [14] for the relation between PDPs and CPDPs). Also, CPDPs can be seen as a generalization of Interactive Markov Chains (IMCs, see [8]). To show the relation of CPDP with IMC, we describe in Section 2 how the CPDP model originated from the IMC model. This section ends with a formal definition of the CPDP model.

CPDPs are designed for communication/interaction with other CPDPs. In Section 3 we describe how CPDPs can be interconnected by using so called parallel composition operators. The use of these parallel composition operators is very common in the field of process algebra (see for example [11] and [9]). We make use of the active/passive composition operators from [13]. We show how composition of CPDPs originates from composition of IMCs. We state the result that the result of composing two CPDPs is again a member of the class of CPDPs. This means that the behavior of two (or more) simultaneously evolving CPDPs, which communicate with each other, can be expressed as a single CPDP. In this way, a complex CPDP can be modelled in a compositional way by modelling its components (as CPDPs) and by selecting the right composition operators to interconnect the component-CPDPs.

Section 4 concerns the relation between CPDPs and PDPs. A PDP is a stochastic process. The behavior of a CPDP can in general not be described by a stochastic process because 1. a CPDP can have multiple hybrid jumps (i.e. the hybrid state discontinuously jumps to another hybrid state) at the same time instant and 2. a CPDP can have nondeterminism, which means that certain choices that influence the state evolution are unmodelled instead of probabilistic as in PDPs. In order to guarantee that the state evolution of a CPDP can be modelled by a stochastic process (and can then be stochastically analyzed), we introduce the concept of scheduler. A scheduler can be seen as a supervisor, which makes probabilistic choices to resolve non-determinism of the CPDP). Then we give an algorithm to check whether a CPDP with scheduler can be converted into a CPDP (with scheduler) that has only one hybrid jump per time instant (i.e. hybrid jumps of multiplicity greater than one are converted to hybrid jumps of multiplicity one). Finally we show that the evolution of the state of a CPDP with scheduler, whose hybrid jumps all have multiplicity one, can be modelled as a PDP. The contents of this section are based on [5]).

In Section 5, we enrich the communication mechanism of CPDPs with so called value passing. With this notion of value passing, a CPDP can receive information about the output variables of other CPDPs. The enriched framework is called value-passing CPDPs. Value-passing is a concept that is successfully used for several process algebra models (see for example [1] and [9] for application of value-passing to the specification language LOTOS). In Section 6 we give an ATM (Air Traffic Management) example of a value passing CPDP. We also apply the algorithm of Section 4 to show that this value-passing CPDP can be converted to a PDP. The ATM-example was first modelled as a Dynamically Coloured Petri Net (DCPN) (see the chapter at pp. 325–350 of this book). DCPN is a Petri net formalism, which has also been designed for compositional specification of PDP-type systems (see [6] and [7] for the DCPN model).

Section 7 is about compositional state reduction by bisimulation. Bisimulation, which we define for CPDP in this section, is a notion of external equivalence. This means that two bisimilar CPDPs cannot be discriminated by an external agent that observes the values of the output variables of the CPDP and interacts with the CPDP. The bisimulation notion that we use is a probabilistic bisimulation (see [10] and [2] for probabilistic bisimulation in the contexts of probabilistic transition systems and probabilistic timed automata). The main result in this section is the bisimulation-substitution-theorem which states that replacing a component of a complex CPDP by another bisimilar component does not change the complex system (up to bisimilarity). In this way we can perform compositional state reduction by reducing the state space of the individual components (via bisimulation). The contents of this section are based on [15]).

The chapter ends in Section 8 with conclusions and a small discussion on compositional modelling and analysis in the context of stochastic hybrid systems.

2 The CPDP Model

In this section we describe how the CPDP model originates from the IMC model. We start with describing the IMC model.

2.1 Interactive Markov Chains

An IMC (Interactive Markov Chain) is a quadruple $(L, \Sigma, \mathcal{A}, \mathcal{S})$, where L is the set of locations (or discrete states), Σ is the set of actions (or events), $\mathcal{A}$ is the set of interactive transitions and consists of triples (l, a, l') with $l, l' \in L$ and $a \in \Sigma$, and $\mathcal{S}$ is the set of Markovian (or spontaneous) transitions and consists of triples (l, λ, l') with $l, l' \in L$ and $\lambda \in \mathbb{R}^+$.

In Figure 1 we see an IMC with two locations, l_1 and l_2, with two interactive transitions (pictured as solid arrows) labelled with event a and with

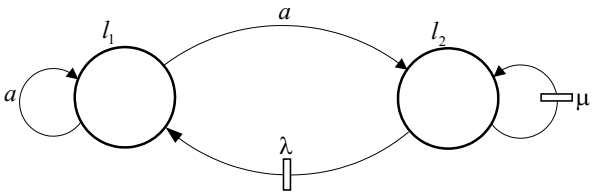

Fig. 1. Interactive Markov Chain

two Markovian transitions (pictured as solid arrows with a little box) labelled with rates λ and μ.

The semantics of the IMC of Figure 1 is as follows: suppose that l_1 in Figure 1 is the initial location (at time $t = 0$). Two things can happen: either the interactive transition labelled a from l_1 to l_2 is taken, or the interactive transition labelled a from l_1 to itself is taken. Note that the choice between these two transitions is not modelled in the IMC, is not determined by the IMC, therefore non-determinism is present at this point (later we will call this form internal non-determinism). Also the time when one of the a-transitions is taken is not modelled (and is therefore left non-deterministic). Suppose that at some time t_1 the a-transition to l_2 is taken. Then at the same time t_1 the process arrives in l_2 (i.e. transitions do not consume time). In l_2 there are two possibilities: either the Markovian transition from l_2 to l_1 with rate λ is taken or the Markovian transition from l_2 to itself with rate μ is taken. In this case neither the choice between these two transitions nor the time of the transition is non-deterministic. The choice and the time are determined probabilistically by a race of Poisson processes: as soon as the process arrives in l_2, two Poisson processes are started with constant rates λ and μ. The process that generates the first point then determines the time and the transition to be taken. Recall that the probability density function of the time of the first point generated by a Poisson process with constant rate λ is equal to $\lambda e^{-\lambda t}$. Suppose that the Poisson process of the λ-transition generates a point after one second and that the Poisson process of the μ-transition generates a point after two seconds, then at time $t = t_1 + 1$ the λ transition is taken which brings the process back to l_1.

2.2 From IMC to CPDP

The first step we could take for transforming the IMC model into the CPDP model is assigning continuous dynamics to the locations. If, in Figure 1, we assign the input/output system $\dot{x} = f_1(x)$,$y = g_1(x)$, with x and y taking value in $\mathbb{R}$ and f_1 and g_1 continuous mappings from $\mathbb{R}$ to $\mathbb{R}$, to l_1 and we assign $\dot{x} = f_2(x)$,$y = g_2(x)$ (with x and y of the same dimensions as x and y of l_1) to l_2, then the resulting process can be pictured as in Figure 2

Suppose that the input/output systems of l_1 and l_2 have given initial states x_1 and x_2 respectively. Then the semantics of the process of Figure 2 would

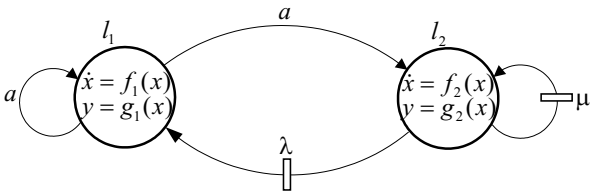

Fig. 2. Interactive Markov Chain enriched with continuous dynamics

be the same as the process of Figure 1, except that when the process is in l_1, then there are continuous variables x and y evolving according to f_1 and g_1 and when the process jumps to l_2, variable x is reset to x_2 (the initial continuous state of l_2) and x and y will then evolve according to f_2 and g_2.

So far, there is little interaction between the discrete dynamics (i.e. the transitions) and the continuous dynamics (i.e. the input/output systems). The transitions are executed independently of the (values of the) continuous variables. The evolution of the continuous variables depends on the transitions as far as it concerns the reset: after every transition, the state variable x is reset to a given value.

In the field of Hybrid Systems, the systems that are studied typically do have (much) interaction between the discrete and the continuous dynamics. In the next step towards the CPDP model, we add some of these interaction possibilities to the model of Figure 2: we add guards, we add reset maps and we allow that the (Poisson) rate of Markovian transitions depends on the value of the continuous variables (and might therefore be non-constant in time).

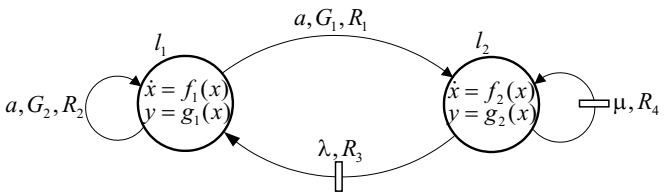

Fig. 3. Interactive Markov Chain enriched with continuous dynamics and discrete/continuous interaction

Guards

We add a guard to each interactive transition. In Figure 3, G_1 and G_2 are the guards. We define a guard of a transition α as a subset of the continuous state space of the origin location of α. In Figure 3 the origin location of the a-transition from l_1 to l_2, is l_1 and therefore G_1 is a subset of $\mathbb{R}$, which is the state space of x at location l_1. The meaning of guard G_1 is that the a-transition to l_2 may not be executed when the value of x (at location l_1) does

not lie in G_1 and it may be executed when $x \in G_1$. Via the guards, interactive transitions depend on the continuous variables.

Reset maps

We add reset maps to each interactive and each Markovian transition. A reset map of a transition α probabilistically resets the value of the state of the target location of α, at the moment that α is executed. Therefore, a reset map is a probability measure on the state space of the target location. We also allow to have different (reset) probability measures for different values of the state variables just before the transition is taken. Suppose that the a-transition to l_2 is taken at the moment that the variable x (at l_1) equals $\hat{x}$. Then $R_1(\hat{x})$ is a probability measure that chooses the new value of x at l_2.

Poisson jump rates

We let Poisson jump rates of a Markovian transition depend (continuously) on the state value of the origin location. In Figure 3, λ, whose transition has origin location l_2, is thus a function from $\mathbb{R}$ (the state space of l_2) to $\mathbb{R}$. If $\lambda(\hat{x}_1) > \lambda(\hat{x}_2)$, then this can be interpreted as: the probability that the Poisson process (corresponding to λ) generates a point within a small time interval when $x = \hat{x}_1$ is bigger than the probability of the generation of a point within the same small time interval when $x = \hat{x}_2$. Suppose that (for example after the a-transition from l_1) x in l_2 is at time t_1 reset to $\hat{x}$. Let $x(t)$ (with $x(t_1) := \hat{x}$) be the value of variable x at time t when x evolves along the vectorfield f_2. Then, the probability density function of the time of the first point generated by the Poisson process with rate $\lambda(x(t))$ is equal to $\lambda(x(t))\mathrm{e}^{-\int_0^t \lambda(x(s))\mathrm{d}s}$.

2.3 Interaction Between Concurrent Processes

The generality of the model of Figure 3 is in fact the generality that we want as far as it concerns the modelling of non-composite systems (i.e. systems that consist of only one component). However, the main aim of the modelling framework that we develop, is compositional modelling. A framework is suitable for compositional modelling if it is possible to model each component of the (composite) system separately and interconnect these separate component-models such that the result describes the behavior of the composite system. With components of a system we mean parts of the system that are running/working simultaneously. For example an Air Traffic Management system that includes multiple (flying) aircraft, where each aircraft forms one subsystem, consists (partly) of subsystems (or components) that 'run' simultaneously. In many composite systems, the components are not independent of each another, but are able to interact with each other and consequently to influence each other. In an ATM system, one aircraft might

send a message (via radio) to another aircraft, which might change the course of the aircraft that receives the message. This is a broadcasting kind of interaction/communication, where there is a clear distinction between the active partner (the one that sends the message) and the passive partner (the one that receives the message). We want to add the possibility of broadcasting communication to the model of Figure 3. In order to do so, we add another type of transition to the model called *passive transitions*. This addition brings us to the class of CPDPs (Communicating Piecewise Deterministic Markov Processes), which will be formally defined after the next paragraph.

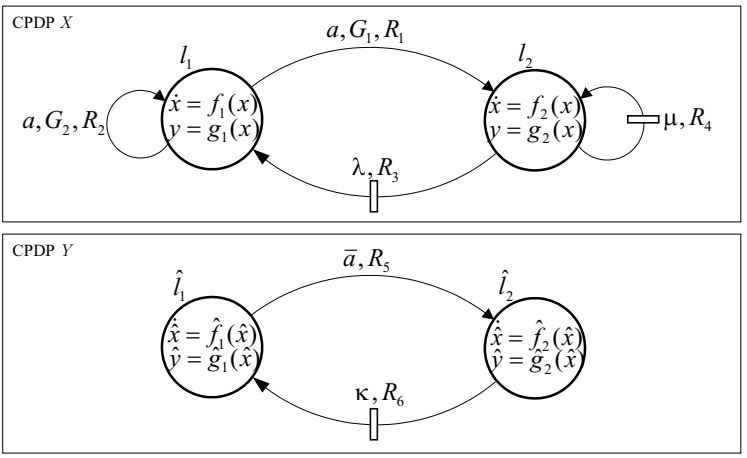

Fig. 4. Two CPDP automata. CPDP Y has a passive transition with label $\bar{a}$.

In Figure 4 we see two CPDPs. CPDP X is the one from Figure 3 and does not have passive transitions. CPDP Y has a passive transition from $\hat{l}_1$ to $\hat{l}_2$ and has a spontaneous transition from $\hat{l}_2$ to $\hat{l}_1$. The passive transition is pictured as a solid arrow, the bar on top of the event label ($\bar{a}$ in Figure 4) denotes that the event is a passive event and that the transition is therefore a passive transition. The passive transition with event $\bar{a}$ reflects that the message a is received. A message a can only be received if some other CPDP has broadcast a message a. Now we can interpret the label a above an interactive transition as: if this transition is executed, the message a is broadcast. We assume that broadcasting and receiving of a message happens instantly (i.e. does not consume time).

For CPDPs, we use the term *active transition* instead of the IMC term *interactive transition* to stress the distinction between activeness and passiveness of transitions. The CPDP terminology for *Markovian transition* is *spontaneous transition*.

2.4 Definition of CPDP

We now give the formal definition of CPDP as an automaton.

Definition 1. *A CPDP is a tuple* $(L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$*, where*

- L *is a set of locations*
- V *is a set of state variables. With* $d(v)$ *for* $v \in V$ *we denote the dimension of variable* v*.* $v \in V$ *takes its values in* $\mathbb{R}^{d(v)}$*.*
- W *is a set of output variables. With* $d(w)$ *for* $w \in W$ *we denote the dimension of variable* w*.* $w \in W$ *takes its values in* $\mathbb{R}^{d(w)}$*.*
- $\nu : L \to 2^V$ *maps each location to a subset of* V*, which is the set of state variables of the corresponding location.*
- $\omega : L \to 2^W$ *maps each location to a subset of* W*, which is the set of output variables of the corresponding location.*
- F *assigns to each location* l *and each* $v \in \nu(l)$ *a mapping from* $\mathbb{R}^{d(v)}$ *to* $\mathbb{R}^{d(v)}$*, i.e.* $F(l, v) : \mathbb{R}^{d(v)} \to \mathbb{R}^{d(v)}$*.* $F(l, v)$ *is the vector field that determines the evolution of* v *for location* l *(i.e.* $\dot{v} = F(l, v)$ *for location* l*).*
- G *assigns to each location* l *and each* $w \in \omega(l)$ *a mapping from* $\mathbb{R}^{d(v_1)+\cdots+d(v_m)}$ *to* $\mathbb{R}^{d(w)}$*, where* v_1 *till* v_m *are the state variables of location* l*.* $G(l, w)$ *determines the output equation of* w *for location* l *(i.e.* $w = G(l, w)$*).*
- Σ *is the set of communication labels.* $\bar{\Sigma}$ *denotes the 'passive' mirror of* Σ *and is defined as* $\bar{\Sigma} = \{\bar{a} | a \in \Sigma\}$*.*
- $\mathcal{A}$ *is a finite set of active transitions and consists of five-tuples* (l, a, l', G, R)*, denoting a transition from location* $l \in L$ *to location* $l' \in L$ *with communication label* $a \in \Sigma$*, guard* G *and reset map* R*.* G *is a closed subset of the state space of* l*. The reset map* R *assigns to each point in* G *for each variable* $v \in \nu(l')$ *a probability measure on the state space (and its Borel sets) of* v *for location* l'*.*
- $\mathcal{P}$ *is a finite set of passive transitions of the form* $(l, \bar{a}, l', R)$*.* R *is defined on the state space of* l *(as the* R *of an active transition is defined on the guard space).*
- $\mathcal{S}$ *is a finite set of spontaneous transitions and consists of four-tuples* (l, λ, l', R)*, denoting a transition from location* $l \in L$ *to location* $l' \in L$ *with jump-rate* λ *and reset map* R*. The jump rate* λ *(i.e. the Poisson rate of the Poisson process of the spontaneous transition) is a mapping from the state space of* l *to* $\mathbb{R}_+$*.* R *is defined on the state space of* l *as it is done for passive transitions.*

Example 1. CPDP X of Figure 4 is defined as:

$(L_X, V_X, \nu_X, W_X, \omega_X, F_X, G_X, \Sigma, \mathcal{A}_X, \mathcal{P}_X, \mathcal{S}_X)$ with $L_X = \{l_1, l_2\}$, $V_X = \{x\}$, $\nu_X(l_1) = \nu_X(l_2) = \{x\}$, $W_X = \{y\}$, $\omega_X(l_1) = \omega_X(l_2) = \{y\}$, $F_X(l_1, x) = f_1(x)$ and $F_X(l_2, x) = f_2(x)$, $G_X(l_1, x) = g_1(x)$ and $G_X(l_2, x) = g_2(x)$, $\Sigma = \{a\}$, $\mathcal{A}_X = \{(l_1, a, l_2, G_1, R_1), (l_1, a, l_1, G_2, R_2)\}$,$\mathcal{P}_X = \emptyset$, $\mathcal{S}_X = \{(l_2, \lambda, l_1, R_3), (l_2, \mu, l_2, R_4)\}$. CPDP Y of Figure 4 is defined as:

$(L_Y, V_Y, \nu_Y, W_Y, \omega_Y, F_Y, G_Y, \Sigma, \mathcal{A}_Y, \mathcal{P}_Y, \mathcal{S}_Y)$ with $L_Y = \{\hat{l}_1, \hat{l}_2\}$, $V_Y = \{\hat{x}\}$, $\nu_Y(\hat{l}_1) = \nu_Y(\hat{l}_2) = \{\hat{x}\}$, $W_Y = \{\hat{y}\}$, $\omega_Y(\hat{l}_1) = \omega_Y(\hat{l}_2) = \{\hat{y}\}$, $F_Y(\hat{l}_1, \hat{x}) = \hat{f}_1(\hat{x})$ and $F_Y(\hat{l}_2, \hat{x}) = \hat{f}_2(\hat{x})$, $G_Y(\hat{l}_1, \hat{x}) = \hat{g}_1(\hat{x})$ and $G_Y(\hat{l}_2, \hat{x}) = \hat{g}_2(\hat{x})$, $\Sigma = \{a\}$, $\mathcal{A}_Y = \emptyset$,$\mathcal{P}_Y = \{(\hat{l}_1, \bar{a}, \hat{l}_2, R_5)\}$, $\mathcal{S}_Y = \{(\hat{l}_2, \kappa, \hat{l}_1, R_6)\}$.

For a CPDP X with $v \in V_X$, where V_X is the set of state variables of X, we call $\mathbb{R}^{d(v)}$ the state space of state variable v. We call $\{(v = r) | r \in \mathbb{R}^{d(v)}\}$ the valuation space of v and each $(v = r)$ for $r \in \mathbb{R}^{d(v)}$ is called a valuation. We call $\{(v_1 = r_1, v_2 = r_2, \cdots, v_m = r_m) | r_i \in \mathbb{R}^{d(v_i)}\}$, where v_1 till v_m are the variables from $\nu(l)$, the valuation space or state space of location l and each $(v_1 = r_1, \cdots, v_m = r_m)$ is called a valuation or state of l. A valuation (state) is an unordered tuple (e.g. $(v_1 = 0, v_2 = 1)$ is the same valuation as $(v_2 = 1, v_1 = 0)$). We denote the valuation space of l by $val(l)$. We call $\{(l, x) | l \in L, x \in val(l)\}$ the state space of a CPDP with location set L and valuation spaces $val(l)$. Each state of a CPDP consists of a location (which comes from a discrete set) and a valuation (which comes from a continuum), therefore we call the state (state space) of a CPDP also hybrid state (hybrid state space). The state space of a location l with $\nu(l) = \{v_1, \cdots, v_m\}$ can be seen as $\mathbb{R}^{d(v_1)+\cdots+d(v_m)}$, because the state space is (topologically) homeomorphic to $\mathbb{R}^{d(v_1)+\cdots+d(v_m)}$ with homeomorphism $\pi_l : val(l) \to \mathbb{R}^{d(v_1)+\cdots+d(v_m)}$ with $\pi_l((v_1 = r_1, \cdots, v_m = r_m)) = (r_1, \cdots, r_m)$. We use unordered tuples for the valuations (states) because this will turn out to be helpful for the composition operation and for some other definitions and proofs.

3 Composition of CPDPs

In the process algebra and concurrent processes literature it is common to define a *parallel composition operator*, normally denoted by $||$. $||$ has as its arguments two processes, say X and Y, of a certain class of processes. The result of the composition operation, denoted by $X||Y$, is again a process that falls within the same class of processes (i.e. the specific class of processes is closed under $||$). The main idea of using this kind of composition operator is that the process $X||Y$ describes the behavior of the composite system that consists of components X and Y (which might interact with each other).

3.1 Composition for IMCs

The interaction-mechanism used for IMCs (see [8]) is not broadcasting interaction but is *interaction via shared events*. This means that if X and Y are two interacting IMCs and a is (by definition) a shared event, then an interactive a-transition of X can only be executed when at the same time an a-transition of Y is executed (and vice versa). In other words, an a-transition of X has to synchronize with an a transition of Y (and vice versa). Markovian

transitions, and interactive transitions with labels that are (by definition) not shared events, can be executed independently of the other component. This notion of interaction for IMC is formalized by a parallel composition operator. If we define A as the set of shared events and we denote the corresponding IMC composition operator by $||_A$, then $||_A$ is defined as follows:

Definition 2. *Let $X = (L_X, \Sigma, \mathcal{A}_X, \mathcal{S}_X)$ and $Y = (L_Y, \Sigma, \mathcal{A}_Y, \mathcal{S}_Y)$ be two IMCs, having the same set of events. Let $A \subset \Sigma$ be the set of shared events. Then $X||_A Y$ is the IMC $(L, \Sigma, \mathcal{A}, \mathcal{S})$, where $L := \{l_1||_A l_2 \mid l_1 \in L_X, l_2 \in L_Y\}$ and where $\mathcal{A}$ and $\mathcal{S}$ are the smallest sets that satisfy the following (structural operational) composition rules:*

$$1.\frac{l_1 \xrightarrow{a} l_1', l_2 \xrightarrow{a} l_2'}{l_1||_A l_2 \xrightarrow{a} l_1'||_A l_2'}(a \in A), \tag{1}$$

$$2a.\frac{l_1 \xrightarrow{a} l_1'}{l_1||_A l_2 \xrightarrow{a} l_1'||_A l_2}(a \notin A), 2b.\frac{l_2 \xrightarrow{a} l_2'}{l_1||_A l_2 \xrightarrow{a} l_1||_A l_2'}(a \notin A), \tag{2}$$

$$3a.\frac{l_1 \xrightarrow{\lambda} l_1'}{l_1||_A l_2 \xrightarrow{\lambda} l_1'||_A l_2}, 3b.\frac{l_2 \xrightarrow{\lambda} l_2'}{l_1||_A l_2 \xrightarrow{\lambda} l_1||_A l_2'}. \tag{3}$$

Here, $l_1 \xrightarrow{a} l_1'$ means $(l_1, a, l_1') \in \mathcal{A}_X$, $l_2 \xrightarrow{a} l_2'$ means $(l_2, a, l_2') \in \mathcal{A}_Y$, $l_1 \xrightarrow{\lambda} l_1'$ means $(l_1, \lambda, l_1') \in \mathcal{S}_X$, $l_2 \xrightarrow{\lambda} l_2'$ means $(l_2, \lambda, l_2') \in \mathcal{S}_Y$, $l_1||l_2 \xrightarrow{a} l_1'||l_2'$ means $(l_1||l_2, a, l_1'||l_2') \in \mathcal{A}$, $l_1||l_2 \xrightarrow{\lambda} l_1'||l_2$ means $(l_1||l_2, \lambda, l_1'||l_2) \in \mathcal{S}$, etc. Furthermore, $\frac{B}{C}(A)$ should be read as "If A and B, then C", and $\frac{B_1, B_2}{C}(A)$ should be read as: if A and B_1 and B_2, then C.

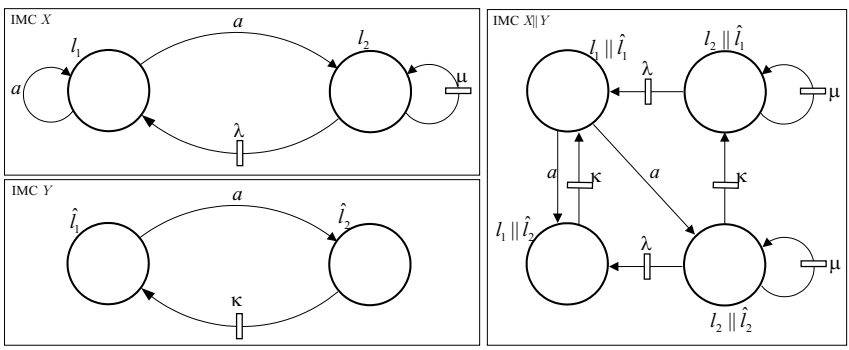

Fig. 5. Composition of two IMCs

In Figure 5, we see on the left two IMCs, X and Y, and we see on the right the IMC $X||Y$, where $||$ is used as shorthand notation for $||_{\{a\}}$. We now check that indeed $X||Y$ expresses the combined behavior of IMCs X and Y

interacting on shared event a: suppose that X and Y initially start in locations l_1 and $\hat{l}_1$ respectively. In $X||Y$, this joint initial location is represented by the location named $l_1||\hat{l}_1$. For a transition to be executed, there are two possibilities: 1. X takes the a transition to l_1 while Y at the same time takes the a-transition to $\hat{l}_2$, 2. X takes the a transition to l_2 while Y at the same time takes the a-transition to $\hat{l}_2$. Note that, since a is a shared event, it is not possible that X takes an a-transition, while Y idles (i.e. stays in location $\hat{l}_1$). Case 1 and 2 are in $X||Y$ represented by the a-transitions to locations $l_1||\hat{l}_2$ and $l_2||\hat{l}_2$ respectively. Note that in cases 1 and 2 one a-transition in $X||Y$ reflect two combined (or synchronized) transitions, one in X and one in Y. If case 2 is executed, then right after the synchronized a-transitions (of X and Y) three Poisson processes are started. Two from X (with parameters λ and μ) and one from Y (with parameter κ). In $X||Y$ this is reflected by the three Markovian transitions at location $l_2||\hat{l}_2$. Suppose that the λ-process generates the first jump. Then X jumps to location l_1 and Y stays in location $\hat{l}_2$, waiting for the κ-process to generate a jump to location $\hat{l}_1$. In $X||Y$ this is reflected by taking the λ-transition to location $l_1||\hat{l}_2$. Then in location $l_1||\hat{l}_2$ again a Poisson process with parameter κ is started. One could question whether this correctly reflects the behavior of the composite system, because when X jumps to l_1, Y stays in $\hat{l}_2$ and the κ-Poisson process keeps running and is not started again as happens in location $l_1||\hat{l}_2$. That indeed starting the κ-process again reflects correctly the composite behavior is due to the fact that the exponential probability distribution (of the Poisson process) is memoryless, which means that, if R_κ denotes a random variable with exponential distribution function $-\mathrm{e}^{\kappa t}$, then

$$\Pr(R_\kappa > \hat{t} + t | R_\kappa > \hat{t}) = \Pr(R_\kappa > t),$$

where $\Pr(A|B)$ denotes the conditional probability of A given B. We know that when X takes the λ-transition after having spent $\hat{t}$ time units in location l_2, then the κ-process did not generate a jump before $\hat{t}$ time units, i.e. $R_\kappa > \hat{t}$. Therefore it is correct to start the κ process again in location $l_1||\hat{l}_2$. (We will see that the situation for composition of CPDPs will be similar when it comes to restarting Poisson processes after an executed transition). The reader can check that the part of $X||Y$ we did not explain here also correctly reflects the composite behavior of X and Y.

3.2 Composition of CPDPs

We have distinguished two kinds of communication: communication via shared events and communication via active/passive events. For CPDP we want to allow both types of interaction. Some interactions of communicating systems can better be modelled through shared events and some interactions can better be modelled through active/passive events. We refer to [13] for a discussion on this issue. This means that also for two interacting CPDPs, we use a set

A (which is a subset of the set of active events Σ) which contains the events that are used as shared events. Then the active events not in A together with the passive events (i.e. the ones in $\bar{\Sigma}$) can be used for active/passive communication. In Figure 6 we see the CPDP $X||Y$, with $||$ shorthand for $||_\emptyset$ (i.e. we choose to have no shared events for this composition), which reflects the composite behavior of X and Y of Figure 4.

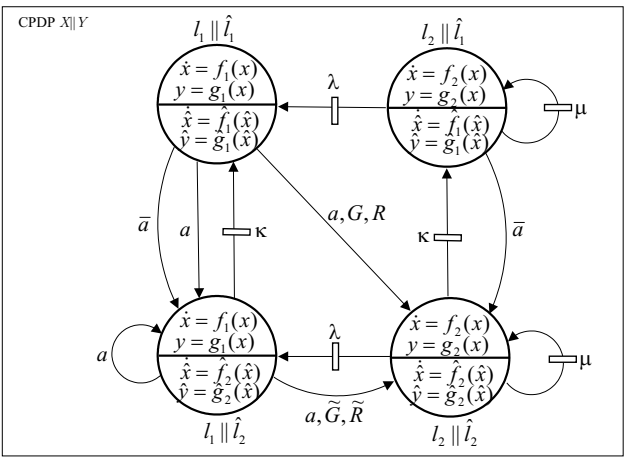

Fig. 6. Composition of two CPDPs (Most guards and reset maps are not drawn)

The communication, reflected by CPDP $X||Y$ of Figure 6, is only through active/passive events (and not through shared events). We will now argue that $X||Y$ of Figure 6 indeed reflects the composite behavior of X and Y interacting via active a and passive $\bar{a}$ events and should therefore be the result of composing X with Y for $A = \emptyset$: suppose X and Y initially start in l_1 and $\hat{l}_1$ respectively, which is reflected by location $l_1||\hat{l}_1$ of $X||Y$. Note that $l_1||\hat{l}_1$ contains the continuous dynamics of both l_1 and $\hat{l}_1$. One possibility is that X executes the a-transition to l_2. Since a is an active event and is not a shared event, X can execute this transition independently of Y. By executing this transition, the message a is send by X. Y has a $\bar{a}$-transition at location $\hat{l}_1$, which means that at $\hat{l}_1$, Y is able to receive the message a. This means that when x executes the a-transition to l_2, Y receives the signal a and synchronizes its $\bar{a}$ transition on the a-transition of X. In Figure 6 this synchronized transition is reflected by the a-transition from $l_1||\hat{l}_1$ to $l_2||\hat{l}_2$. This transition broadcasts signal a which reflects the broadcasting of a by X. $l_1||\hat{l}_1 \xrightarrow{a,G,R} l_2||\hat{l}_2$ (i.e. the a-transition from $l_1||\hat{l}_1$ to $l_2||\hat{l}_2$) can be executed when $x \in G_1$, with G_1 from Figure 4. There is no condition for $\hat{x}$ (i.e. the passive transition can always be taken as soon as an active a-message is broadcast). Therefore G should be equal to $G_1 \times \mathbb{R}^{d(\hat{x})}$. The reset map R should reset x

via R_1 (of Figure 4) and should reset $\hat{x}$ via R_6 (of Figure 4). The probability measures of R_1 and R_6 are independent therefore we can use the product probability measure for $R(x, \hat{x}) = R_1(x) \times R_6(\hat{x})$, where x and $\hat{x}$ are elements from the state spaces of l_1 and $\hat{l}_1$ respectively.

We discuss a few more transitions of $X||Y$:

- $l_1||\hat{l}_2 \xrightarrow{a,\tilde{G},\tilde{R}} l_2||\hat{l}_2$: this transition reflects that X executes the active a-transition to l_2 while Y does not receive the a-message because Y has no $\bar{a}$-transition at location $\hat{l}_2$. Again $\tilde{G}$ should be equal to $G_1 \times \mathbb{R}^{d(\hat{x})}$. $\tilde{R}$ should reset x according to R_1 and should leave $\hat{x}$ unaltered. Therefore $\tilde{R}(x, \hat{x}) = R_1(x) \times Id_{\hat{x}}$, where $Id_{\hat{x}}$ is the identity probability measure for which the set $\{\hat{x}\}$ has probability one (i.e. the probability that $\hat{x}$ stays unaltered after the reset is one).
- $l_1||\hat{l}_2 \xrightarrow{a} l_1||\hat{l}_2$: this transition reflects that X executes $l_1 \xrightarrow{a,G_2,R_2} l_1$ while Y receives no message a. (We do not specify guard and reset map of this transition here).
- $l_2||\hat{l}_2 \xrightarrow{\lambda,\tilde{R}'} l_1||\hat{l}_2$ (reset map $\tilde{R}'$ is not drawn in Figure 6): this transition reflects that X executes the spontaneous λ-transition from l_2 to l_1, while Y stays unaltered. $\tilde{R}'(x, \hat{x})$ should be equal to $R_3(x) \times Id_{\hat{x}}$, with R_3 from Figure 4. Here we have a similar situation as with IMC: after this λ-transition, the κ-process of Y is restarted. As for the IMC case, this is correct because the Poisson process is memoryless. Note that the random variable that belongs to this CPDP κ-process depends on the state where the κ-process is started: if at t_0 the κ-process is activated at state $x(t_0)$ (i.e. a hybrid jump to state $x(t_0)$ took place at time t_0), then the random variable $R_\kappa(x(t_0))$, which denotes the amount of time t after t_0 until κ generates a jump, given that κ is activated at $x(t_0)$, has probability density function $\kappa(x(t_0+t))\mathrm{e}^{-\int_0^t \lambda(x(t_0+s))\mathrm{d}s}$, which is different for different values of t_0. For this situation we get

$$\Pr(R_\kappa(x(t_0)) > \hat{t} + t | R_\kappa(x(t_0)) > \hat{t}) = \Pr(R_\kappa(x(t_0 + \hat{t})) > t),$$

  from which we see that it is correct to (re)activate the κ-process after the transition at state $x(t_0 + \hat{t})$ when it is given that the κ-process that was activated at state $x(t_0)$ did not generate a jump within $\hat{t}$ time units.
- $l_1||\hat{l}_1 \xrightarrow{\bar{a}} l_1||\hat{l}_2$: this transition reflects that Y can also receive a-messages that are not broadcast by X but by some other component Z that we might want to add to the composition $X||Y$. (Then we get the composite model $(X||Y)||Z$).

Because from Figures 4 and 6 we now have an understanding how a CPDP composition operator $||$ should map two CPDPs (X and Y) to a new CPDP ($X||Y$), we are ready to formalize the composition operation. We give a definition of the operator denoted by $|^P_A|$, where A is the set of shared active events and P is the set of shared passive events. So far we did not see the

distinction between shared and non-shared passive events. This distinction is only useful when there are more than two components involved. Suppose we have a composite system with three components. Component one has an active transition with label a and can therefore potentially send the message a. Components two and three both have passive transitions with label $\bar{a}$, therefore they both can potentially receive the message a. Now, if $\bar{a}$ is a shared event of components two and three, then it is possible that both can at the same time receive the signal a of component one (which results into three synchronizing transitions, one active and two passive transitions). If $\bar{a}$ is not a shared event of components two and three, then this means that only one of the components two and three may receive the signal a of component one (i.e. it is not allowed that the three transitions synchronize, only synchronization of one active with one passive transition is allowed). For a discussion on the use of this distinction between shared and non-shared passive events, we refer to [13]. Before we give the definition of composition of CPDPs, we first look at the composition rules (i.e. the operational semantics) of the operator $|^P_A|$.

Suppose we have two CPDPs, X and Y, which interact under the set of shared active events A and the set of shared passive events P. If $a \in A$, then an a-transition in X can be executed only when at the same time an a-transition in Y can be executed. This is expressed by the following composition rule, which is the analogy of the IMC composition rule 1 in (1).

$$r1.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|^P_A|l_2 \overset{a,G_1\times G_2,R_1\times R_2}{\longrightarrow} l_1'|^P_A|l_2'}(a \in A).$$

The synchronized transition, in the CPDP $X|^P_A|Y$, has guard $G_1 \times G_2$, which expresses that if one of the two guards G_1 and G_2 is not satisfied, then the synchronized transition can not be executed. The reset map is constructed via the product probability measures $R_1 \times R_2$, which expresses that R_1 independently resets the state variables of l_1' of X and R_2 independently resets the state variables of l_2' of Y.

If $a \notin A$, then active a-transitions can be executed independently and passive $\bar{a}$-transitions can synchronize on a-transitions of other components. This is expressed by the following composition rule.

$$r2.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|^P_A|l_2 \overset{a,G_1\times val(l_2),R_1\times R_2}{\longrightarrow} l_1'|^P_A|l_2'}(a \notin A).$$

The guard of the synchronized transition equals $G_1 \times val(l_2)$, where $val(l_2)$ denotes the state space of location l_2. This expresses that there is no guard condition on the passive transition (i.e. it may always synchronize when an active a-partner is available). We also need the mirror rule $r2'$:

$$r2'.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|^P_A|l_2 \overset{a,val(l_1)\times G_2,R_1\times R_2}{\longrightarrow} l_1'|^P_A|l_2'}(a \notin A).$$

If $a \notin A$, then an a-transition can be executed also when there is no passive $\bar{a}$-transition available in the other component (A signal can be broadcast also when there is no receiver to receive the message). This is expressed by the following rule r3 and its mirror $r3'$ which we will not explicitly state. The IMC analogy are rules 2a and 2b in (2).

$$r3.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a}}{\not\longrightarrow}}{l_1|_A^P|l_2 \overset{a,G_1\times val(l_2),R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(a \notin A).$$

Here Id is the identity probability measure, which does not change the state value of l_2 with probability one.

The following three rules $r4$,$r5$ and $r6$ concern the passive transitions of $X|_A^P|Y$. A passive $\bar{a}$-transition of $X|_A^P|Y$ reflects that either X or Y can receive an a-message from a component Z that we might want to add to the composition. If $\bar{a} \in P$ and X can execute a $\bar{a}$-transition from location l_1 and Y can execute a $\bar{a}$-transition from location l_2. Then if X is in l_1 and Y is in l_2 and an a-message is broadcast (by the other component Z), then the two passive transitions will be executed at the same time (of the a-message) and will therefore synchronize. This is expressed by the following rule.

$$r5.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{\bar{a},R_1\times R_2}{\longrightarrow} l_1'|_A^P|l_2'}(\bar{a} \in P).$$

If $\bar{a} \in P$, but only one component has a $\bar{a}$-transition to receive the message a from Z, then this component will receive the message while the other component stays unchanged. This is expressed by the following rule $r6$ (and its mirror $r6'$ which we do not explicitly state here).

$$r6.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a}}{\not\longrightarrow}}{l_1|_A^P|l_2 \overset{\bar{a},R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(\bar{a} \in P)$$

If $\bar{a} \notin P$, then two passive $\bar{a}$-transitions cannot synchronize because only one is allowed to receive the message a from Z. Therefore these passive $\bar{a}$-transitions of X and Y remain in the composition (to potentially receive an a-message from Z) but will not synchronize. This is expressed by the following rules $r4$ and $r4'$.

$$r4.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1'}{l_1|_A^P|l_2 \overset{\bar{a},R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(\bar{a} \notin P), \quad r4'.\frac{l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{\bar{a},Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'}(\bar{a} \notin P)$$

Finally we need one more composition rule $r7$ (and its mirror $r7'$) to express that spontaneous transitions of X and Y remain in the composition $X|_A^P|Y$ (as we have seen in the discussion on Figure 6). The IMC analogy of these rules are rules 3a and 3b in (3).

$$r7.\frac{l_1 \xrightarrow{\lambda_1,R_1} l_1'}{l_1|_A^P|l_2 \xrightarrow{\hat{\lambda}_1,R_1\times Id} l_1'|_A^P|l_2}, \quad r7'.\frac{l_2 \xrightarrow{\lambda_2,R_2} l_2'}{l_1|_A^P|l_2 \xrightarrow{\hat{\lambda}_2,Id\times R_2} l_1|_A^P|l_2'}.$$

Here $\hat{\lambda}_1$ and $\hat{\lambda}_2$ are defined on the combined state space of locations l_1 and l_2 and equal $\hat{\lambda}_1(x_1,x_2) = \lambda_1(x_1)$ and $\hat{\lambda}_2(x_1,x_2) = \lambda_2(x_2)$, where x_1 and x_2 are states of l_1 and l_2 respectively.

Definition 3. *If* $X = (L_X, V_X, \nu_X, W_X, \omega_X, F_X, G_X, \Sigma, \mathcal{A}_X, \mathcal{P}_X, \mathcal{S}_X)$ *and* $Y = (L_Y, V_Y, \nu_Y, W_Y, \omega_Y, F_Y, G_Y, \Sigma, \mathcal{A}_Y, \mathcal{P}_Y, \mathcal{S}_Y)$ *are two CPDPs that have the same set of events* Σ *and if we have* $V_X \cap V_Y = W_X \cap W_Y = \emptyset$*, then* $X|_A^P|Y$ *is defined as the CPDP* $(L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$*, where*

- $L = \{l_1|_A^P|l_2 \mid l_1 \in L_X, l_2 \in L_Y\}$,
- $V = V_X \cup V_Y$, $W = W_X \cup W_Y$,
- $\nu(l_1|_A^P|l_2) = \nu(l_1) \cup \nu(l_2)$, $\omega(l_1|_A^P|l_2) = \omega(l_1) \cup \omega(l_2)$,
- $F(l_1|_A^P|l_2, v)$ *equals* $F_X(l_1, v)$ *if* $v \in \nu_X(l_1)$ *and equals* $F_Y(l_2, v)$ *if* $v \in \nu_Y(l_2)$.
- $G(l_1|_A^P|l_2, w)$ *equals* $G_X(l_1, w)$ *if* $w \in \omega_X(l_1)$ *and equals* $G_Y(l_2, w)$ *if* $w \in \omega_Y(l_2)$.
- $\mathcal{A}$, $\mathcal{P}$ *and* $\mathcal{S}$ *contain and only contain the transitions that are the result of applying one of the rules r1,r2,r2',r3,r3',r4,r4',r5,r6,r6',r7 and r7', defined above.*

Example 2. It can be checked that, according to Definition 3, CPDP $X||Y$ from Figure 6 is indeed the resulting CPDP of composing X and Y from Figure 4 with composition operator $|_A^P|$, where $A = \emptyset$ and $P = \bar{\Sigma}$. Note that any other $P \subset \bar{\Sigma}$ would give the same result because X has no passive transitions and therefore it is not relevant for the composition of X and Y whether passive transitions synchronize or not (which is determined by P).

In order to prove that, for certain A and P, the composition operator $|_A^P|$ is commutative and associative, we need to introduce an equivalence notion, that equates CPDPs that are exactly the same except that the locations may have different names. We call this equivalence notion, in the line of [2], *isomorphism* and we define it as follows.

Definition 4. *Two CPDPs* $X = (L_X, V, \nu_X, W, \omega_X, F_X, G_X, \Sigma, \mathcal{A}_X, \mathcal{P}_X, \mathcal{S}_X)$ *and* $Y = (L_Y, V, \nu_Y, W, \omega_Y, F_Y, G_Y, \Sigma, \mathcal{A}_Y, \mathcal{P}_Y, \mathcal{S}_Y)$*, with shared* V,W *and* Σ*, are* isomorphic *if there exists a bijection* $\pi : L_X \to L_Y$ *such that, for all* $l \in L_X$, $\nu_X(l) = \nu_Y(\pi(l))$, $\omega_X(l) = \omega_Y(\pi(l))$, $F_X(l, v) = F_Y(\pi(l), v)$ *for all* $v \in \nu(l)$, $G_X(l, w) = G_Y(\pi(l), w)$ *for all* $w \in \omega(l)$*, for any* a,$\bar{a}$,λ,l', G *and* R *we have that:* $(l, a, l', G, R) \in \mathcal{A}_X$ *if and only if* $(\pi(l), a, \pi(l'), G, R) \in \mathcal{A}_Y$, $(l, \bar{a}, l', R) \in \mathcal{P}_X$ *if and only if* $(\pi(l), \bar{a}, \pi(l'), R) \in \mathcal{A}_Y$, $(l, \lambda, l', R) \in \mathcal{S}_X$ *if and only if* $(\pi(l), \lambda, \pi(l'), R) \in \mathcal{S}_Y$.

We now state a result on the commutativity and associativity of the composition operators $|^P_A|$. The operator $|^P_A|$ is called commutative if for all CPDPs X and Y we have that $X|^P_A|Y$ is isomorphic to $Y|^P_A|X$. The operator $|^P_A|$ is called associative if for all CPDPs X,Y and Z we have that $(X|^P_A|Y)|^P_A|Z$ is isomorphic to $X|^P_A|(Y|^P_A|Z)$.

Theorem 1. *The composition operator $|^P_A|$ is commutative for all A and P. $|^P_A|$ is associative if and only if for all $a \in \Sigma$ we have: if $\bar{a} \notin P$ then $a \in A$.*

Proof. The proof of this theorem in the context of active/passive labelled transition systems can be found on www.cs.utwente.nl/~strubbesn. The proof can easily be generalized to the context of CPDPs.

If we have n CPDPs X_i $(i = 1 \cdots n)$ with events-set Σ that are composed via an associative operator $|^P_A|$, then the order of composition does not influence the resulting CPDP and therefore we can write $X_1|^P_A|X_2|^P_A| \cdots X_{n-1}|^P_A|X_n$ to unambiguously (up to isomorphism) denote the resulting composite CPDP.

4 PDP-Semantics of CPDPs

Under certain conditions, the state evolution of a CPDP can be modelled as a stochastic process. In this section we give the exact conditions under which this is true. We also prove that the stochastic process may always be chosen of the PDP-type. In order to achieve this result, we first need to make a distinction between guarded CPDP states and unguarded CPDP states.

Definition 5. *A state (l, x) of a CPDP X is called* guarded*, if there exists an active transition with origin location l such that x is an element of the guard of this transition. A CPDP state is* unguarded *if it is not guarded.*

If we execute a CPDP X from some initial hybrid state (l_0, x_0) then the first part of the state trajectory (i.e. the evolution of the state variables in time) and of the output trajectory (i.e. the evolution of the output variables in time) is determined by F_X and G_X respectively. This is the case until the first transition is executed, which might cause a jump (i.e. discontinuity) in the state/output trajectories. We choose that at these points of discontinuity, the state/output trajectories have the *cadlag* property, which means that at these points the trajectories are continuous from the right and have limits from the left. If then at $t = t_1$, X executes a transition which resets the state to a unguarded state x_1, then the value of the state trajectory at $t = t_1$ equals x_1 (and the value of the output trajectory equals the output value of x_1). If the state after reset x_1 is guarded, then it is possible that at the same time t_1 from state x_1 another active transition is executed. If this transition resets the state to a unguarded state x'_1, then the value of the state trajectory at t_1 equals x'_1. If this transition resets the state to an guarded state x'_1, then

another active transition can be executed, etc. We see that the CPDP model allows multiple transitions at the same time instant.

Formally, let $E := \{(l, x) | l \in L_X, x \in val(l)\}$ be the state space of CPDP X, where $val(l)$ denotes the space of all valuations for the state variables of location l. The trajectories of X are elements of the space $D_E[0, \infty[$ which is the space of right-continuous E-valued functions on $\mathbb{R}_+$ with left-hand limits. According to [4], a metric can be defined on E such that $(E, \mathcal{B}(E))$, with $\mathcal{B}(E)$ the set of Borel sets of E under this metric, is a Borel space (i.e. a subset of a complete separable metric space) and each Borel set B is such that for each $l \in L_X$, $\{x | (l, x) \in B\}$ (i.e. the restriction of B to l) is a Borel set of the Euclidean state space $val(l)$ of location l. Therefore, the concept of continuity within a location (i.e. for sets $\{(l, x) | x \in val(l)\}$) coincides with the standard (Euclidean) concept of continuity.

The CPDP model exhibits non-determinism. This means that at certain time instants of the execution of a CPDP (from some initial state) choices have to be made which are neither deterministic (like a differential equation deterministically determines (a part of) the state trajectory) nor stochastic (i.e. a probability measure can be used to make a probabilistic choice). These non-deterministic choices are simply unmodelled. We distinguish two sources of non-determinism for the CPDP: 1. The choice when an active transition is taken. 2. The choice which active transition is taken. To resolve non-determinism of type 1, we use, in the line of [8], the *maximal progress* strategy, which means that as soon as the state enters a guard area (i.e. at the first time instant that the state is guarded), an active transition has to be executed. To resolve non-determinism of type 2, we use a socalled *scheduler* S which

1. assigns to each guarded state x a probability measure on the set of all active transitions that have x as an element of their guard (i.e. the set of all active transitions that are allowed to be executed from state x) and
2. assigns to each pair $(x, \bar{a})$, with x any state and $\bar{a} \in \bar{\Sigma}$ such that there is a $\bar{a}$-transition at the location of x, a probability measure on the set of all $\bar{a}$-transitions at the location of x.

In other words, if an active transition has to be executed from state x, S probabilistically chooses which active transition is executed and if an active a triggers a $\bar{a}$-transition, then S probabilistically chooses which $\bar{a}$-transition is executed.

For identifying the stochastic process of a CPDP, we only look at *closed* CPDPs, which are CPDPs that have no passive transitions. Closed CPDPs are called closed because we assume that they represent the whole system (i.e. no more other component-CPDPs will be added). Therefore closed CPDPs should have no passive transitions because passive transitions can only be executed when another component triggers it (via an active transition). The order of finding the stochastic behavior of the composite system is therefore: first compose the different components. Then remove all passive transitions

of the resulting CPDP. This results in a closed CPDP where, under maximal progress and scheduler S, all choices for the execution of the CPDP are made probabilistically. One could question whether the evolution of the state can, for closed CPDPs, be modelled as a stochastic process. We can state a condition on the CPDP under which this is not possible: if with non-zero probability we can reach an guarded state x where with non-zero probability an infinite sequence of active transitions can be chosen such that each transition resets the state within the guard of the next transition, then the trajectory of this execution deadlocks (i.e. time does not progress anymore after reaching x at some time $\hat{t}$ and therefore the trajectory is not defined for time instants after time $\hat{t}$). Trajectories of stochastic processes do not deadlock like this, therefore this state evolution cannot be modelled by a stochastic process.

In order to find the stochastic process of a closed CPDP, we would first like to state decidable conditions on a CPDP, which guarantee that the probability that an execution deadlocks (i.e. comes at a point where time does not progress anymore) is zero.

4.1 The Stochastic Process of a Closed CPDP

Suppose we have a closed CPDP X with location set L_X and active transition set $\mathcal{A}_X$. The CPDP operates under maximal progress and under scheduler S. We write $S_x(\alpha)$ for the probability that active transition α is taken when an active transition is executed at state x. We assume that the CPDP has no spontaneous transitions. The case 'with spontaneous transitions' is treated at the end of this section.

We call the jump of a CPDP from the current state to another unguarded state via a sequence of active transitions a hybrid jump. We call the number of active transitions involved in a hybrid jump the multiplicity of the hybrid jump. For example, if at state x_1 a transition α is taken to x_1', which lies in the guard of transition β, and immediately transition β is taken to a unguarded state x_1'', then this hybrid jump from x_1 to x_1'' has multiplicity two.

We need to introduce the concept of total reset map. $R_{tot}(B, x)$ denotes the probability of jumping into $B \in \mathcal{B}(E)$ when an active jump takes place at state x. We have that

$$R_{tot}(B, x) = \sum_{\alpha \in \mathcal{A}_{l_x \rightarrow}} [S_x(\alpha) R_\alpha(B \cap val(l'_\alpha), x)],$$

where $\mathcal{A}_{l_x \rightarrow}$ is the set of all active transitions that leave the location of x. We define the total guard $G_{tot,l}$ of location l as the union of the guards of all active transitions with origin location l. It can be seen now that for the stochastic executions (i.e. generating trajectories during simulation) of X it is enough to know R_{tot} and $G_{tot,l}$ (for all $l \in L_X$) instead of $\mathcal{A}_X$: a trajectory that starts in (l_0, x_0) evolves until it hits G_{tot,l_0} at some state (l_0, x_1). From x_1 we determine the target state (l_1, x_1') of the (first step of the) hybrid jump

by drawing a sample from $R_{tot}(\cdot, x_1)$. If x_1' is unguarded, the next piecewise deterministic part of the trajectory is determined by the differential equations of the state variables of location l_1 until G_{tot,l_1} is hit. If x_1' is guarded, we directly draw a new target state (l_1', x_1'') from $R_{tot}(\cdot, x_1')$, etc. Therefore, if two closed CPDPs that are isomorphic except for the active transition set, and they have the same total reset map and the same total guards, then the stochastic behaviors (concerning the state trajectories) of the two CPDPs are the same and consequently if some stochastic process models the state evolution of one CPDP, then it also models the state evolution of the other CPDP.

Finding the stable and unstable parts of an active transition

Take any $\alpha \in \mathcal{A}_X$. We now show how to split up α in a stable part α_s and an unstable part α_u such that the stochastic behavior of X does not change.

We define G_{α_s} as the set of all $x \in G_\alpha$ (i.e. all x in the guard of α) such that $R_\alpha(val_s(l'_\alpha), x) \neq 0$, where $val_s(l'_\alpha)$ is the unguarded part of the state space of the target location of α. Then for all $x \in G_{\alpha_s}$ we define

$$R_{\alpha_s}(B, x) := \frac{R_\alpha(B \cap val_s(l'_\alpha), x)}{R_\alpha(val_s(l'_\alpha), x)},$$

$$S_x(\alpha_s) := S_x(\alpha) R_\alpha(val_s(l'_\alpha), x).$$

The scheduler works on α_s as $S_x(\alpha_s)$ (as defined above).

We define G_{α_u} as the set of all $x \in G_\alpha$ such that $R_\alpha(val_u(l'_\alpha), x) \neq 0$. For all $x \in G_{\alpha_u}$ we define

$$R_{\alpha_u}(B, x) := \frac{R_\alpha(B \cap val_u(l'_\alpha), x)}{R_\alpha(val_u(l'_\alpha), x)},$$

$$S_x(\alpha_s) := S_x(\alpha) R_\alpha(val_u(l'_\alpha), x).$$

The scheduler works on α_u as $S_x(\alpha_u)$ (as defined above).

It can be seen that replacing α by α_s and α_u does not change the total reset map.

Resolving hybrid jumps of multiplicity greater than one

For any $n \in \mathbb{N}$ we will now define T_s^n and T_u^n. T_s^n is a set of stable transitions representing hybrid jumps of multiplicity n and T_u^n is a set of unstable transitions representing hybrid jumps of multiplicity n. A stable transition is a transition that always jumps to the unguarded state space of the target location. An unstable transition always jumps to the guarded state space. A stable transition is stable in the sense that after the hybrid jump caused by the transition, no other hybrid jump will happen immediately and therefore we are sure that a stable transition will not cause an explosion of hybrid jumps

(i.e. a hybrid jump of multiplicity infinity). An unstable transition does not need to induce such a blow up of hybrid jumps, but potentially it can.

We define T_s^1 as the set of all active transitions α_s (with $\alpha \in \mathcal{A}_X$) such that $G_{\alpha_s} \neq \emptyset$ and we define T_u^1 as the set of all active transitions α_u (with $\alpha \in \mathcal{A}_X$) such that $G_{\alpha_u} \neq \emptyset$.

We introduce the following notations. $P_x(B \circ \beta \circ \alpha)$ denotes the probability that, given that an active jump takes place at state x, transition α is executed followed directly by transition β jumping into the set $B \in \mathcal{B}(val(l'_\beta))$. It can be seen that

$$P_x(B \circ \beta \circ \alpha) = S_x(\alpha) \int_{\hat{x} \in G_\beta} S_{\hat{x}}(\beta) R_\beta(B, \hat{x}) \mathrm{d}R_\alpha(\hat{x}, x).$$

We will now inductively determine the sets T_s^n and T_u^n. Suppose the sets T_s^{n-1} and T_u^{n-1} and T_s^1 and T_u^1 are given. Now, for any $\alpha \in T_u^{n-1}$, $\beta \in T_s^1 \cup T_u^1$ such that $l'_\alpha = l_\beta$, we define $G_{\beta \circ \alpha}$ as all $x \in G_\alpha$ such that $R_\alpha(G_\beta, x) \neq 0$. Then, for all $x \in G_{\beta \circ \alpha}$ we define

$$S_x(\beta \circ \alpha) := P_x(val(l'_\beta) \circ \beta \circ \alpha),$$

$$R_{\beta \circ \alpha}(B, x) := \frac{P_x(B \circ \beta \circ \alpha)}{S_x(\beta \circ \alpha)}.$$

If $G_{\beta \circ \alpha} \neq \emptyset$ and $\beta \in T_s^1$ then we add transition $\beta \circ \alpha$, with guard, reset map and scheduler as above, to T_s^n. If $G_{\beta \circ \alpha} \neq \emptyset$ and $\beta \in T_u^1$ then we add transition $\beta \circ \alpha$, with guard, reset map and scheduler as above, to T_u^n.

Finding the PDP that models the state evolution of the CPDP

If we define, for $z \in \{s, u\}$ and $B \in \mathcal{B}(E)$,

$$R_{tot,z}^n(B, x) := \sum_{\{\alpha \in T_z^n | l_\alpha = l_x\}} [S_x(\alpha) R_\alpha(B \cap val(l'_\alpha), x)],$$

with $B \cap val(l'_\alpha)$ sloppy notation for $\{x | x \in val(l'_\alpha), (l'_\alpha, x) \in B\}$, then it can be seen that for any $n \in \mathbb{N}$ we have

$$R_{tot}(B, x) = \sum_{i=1}^{n} [R_{tot,s}^i(B, x)] + R_u^n(B, x),$$

with other words, if X^n is isomorphic to CPDP X, except that the active transition set of X^n equals $T_s^1 \cup T_s^2 \cup \cdots \cup T_s^n \cup T_u^n$ (which need not be isomorphic to $\mathcal{A}_X$), then the total reset maps of X and X^n are the same for all n.

We are now ready to state the theorem which gives necessary and sufficient conditions on the CPDP such that the state evolution can be modelled by a stochastic process. Also, the theorem says that if the state evolution can be modelled by a stochastic process, then it can be modelled by a stochastic process from the class of PDPs. The proof of the theorem makes use of the results from [14].

Theorem 2. *Let X^n be derived from X as above. Let $R^n_{tot,s}$ denote the total stable reset map of X^n. The state evolution of X can be modelled by a stochastic process if and only if $R(E,x) := \lim_{n\to\infty} R^n_{tot,s}(E,x) = 1$ for all $x \in E_u$, with E_u the guarded part of E. If this condition is satisfied, then the PDP with the same state space as X, with invariants $E^0_l = val(l)\backslash G_{tot,l}$ and with transition measure $Q(B,x) = R(B,x)$, models the state evolution of X.*

Proof. From the text above and from the results of [14], it is clear that if $R(E,x) = 1$ for all x, then the PDP suggested by the theorem models the state evolution of X. If for some $x \in E$, $R(E,x) < 1$, then it can be seen that this must mean that there exists a hybrid jump with multiplicity infinity such that the probability of this hybrid jump at x is greater than zero. This means that (from x) there is a deadlock probability (i.e. time does not progress anymore) greater than zero, which means that the state evolution of X cannot be modelled by a stochastic process (as we saw before).

Corollary 1. *If for some $n \in N$ we have that $T^n_u = \emptyset$, then the multiplicity of the hybrid jumps of X is bounded by n and the state of X exhibits a PDP behavior, with the same PDP as the corresponding PDP of X^n (which can be constructed according to [14] because all hybrid jumps of X^n have multiplicity one).*

The case including spontaneous transitions

Now we treat the case where there are also spontaneous transitions present. Let X be a CPDP without passive and spontaneous transitions and let $\hat{X}$ be an isomorphic copy of X together with a set of spontaneous transitions $\mathcal{S}_{\hat{X}}$. Suppose that the multiplicity of the hybrid jumps of X is bounded by n. Let $\hat{X}^n$ be an isomorphic copy of X^n together with the following spontaneous transitions: for any spontaneous transition $(l,\lambda,l',R) \in \mathcal{S}_{\hat{X}}$ we add to $\hat{\mathcal{S}}$, which denotes the set of spontaneous transitions of $\hat{X}^n$, the transition $(l,\lambda,L,\hat{R})$, where, for $B \in \mathcal{B}(E)$, $\hat{R}(B,x) :=$

$$R(B \cap Inv_s(l'),x) + \sum_{\{\alpha \in \mathcal{A}_{X^n} | l_\alpha = l\}} \int_{\hat{x}\in G_\alpha} S_{\hat{x}}(\alpha) R_\alpha(B \cap val(l'_\alpha)) \mathrm{d}R(\hat{x},x).$$

Note that all transitions from $\mathcal{A}_{X^n}$ are stable. Also note that $(l,\lambda,L,\hat{R})$ is not a standard CPDP transition, but a transition that represents a Poisson process in location l with jump-rate λ and with reset map $\hat{R}$, which can jump to multiple locations. Therefore we write L instead of l' in the tuple of the transition.

It is known that the superposition of two (or more) Poisson processes is again a Poisson process (see, in the context of CPDP, [14] for a proof of this result). This means that if we combine all spontaneous transitions of $\hat{X}^n$ with origin location l to one spontaneous transition $(l,\lambda_l,L,\hat{R}_{tot,l})$, with

$$\lambda_l(x) = \sum_{\alpha \in \hat{\mathcal{S}}_{l\rightarrow}} \lambda_\alpha(x),$$

and

$$\hat{R}_{tot,l}(B,x) = \sum_{\alpha \in \hat{\mathcal{S}}_{l\rightarrow}} [\frac{\lambda_\alpha(x)}{\lambda_l(x)} R_\alpha(B,x)],$$

and if we replace all spontaneous transitions by these combined spontaneous transitions, then the stochastic behavior (concerning the evolution of the state) will not change. Now it can be easily seen that if we add jump rate $\lambda(l,x) = \lambda_l(x)$ to the PDP that models the state evolution of X and we let, for unguarded states (l,x), the transition measure $Q(B,(l,x)) = \hat{R}_{tot,l}(B,x)$, then this PDP will model the state evolution of $\hat{X}$.

5 Value-Passing CPDPs

In the CPDP-model as it is defined so far, it is not possible that one component can inform another component about the value of its state or output variables. In Dynamically Colored Petri Nets (see [6]), this is possible. In this section we introduce an addition to the CPDP model, which adds this feature of communicating state data. We chose to follow a standard method of data communication, called *value-passing*. Value-passing has been defined for different models like LOTOS ([9]). Value-passing can be seen as a natural extension to (the standard) communication through shared events because it is also expressed through "shared events"/"synchronization of active transitions".

5.1 Definition of Value-Passing CPDP

We introduce a new definition for CPDP, which makes communication of state data possible.

Definition 6. *A value-passing CPDP is a tuple* $(L, V, W, \nu, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$*, where all elements except* $\mathcal{A}$ *are defined as in Definition 1 and where* $\mathcal{A}$ *is a finite set of active transitions that consists six-tuples* (l, a, l', G, R, vp)*, denoting a transition from location* $l \in L$ *to location* $l' \in L$ *with communication label* $a \in \Sigma$*, guard* G*, reset map* R *and value-passing element* vp*.* G *is a subset of the state space of* l*.* vp *can be equal to either* $!Y$*,* $?U$ *or* $\emptyset$*. For the case* $!Y$*,* Y *is an ordered tuple* $(w_1, w_2, \cdots, w_m)$ *where* $w_i \in w(l)$ *for* $i = 1 \cdots m$*, meaning that this transition can pass the values of the variables from* Y *(in this specific order) to other transitions in other components. For the case* $?U$*, we have* $U \subset \mathbb{R}^n$ *for some* $n \in \mathbb{N}$*, meaning that this transition asks for input a tuple of the form of* Y *with total dimension* n *(i.e.* $\sum_{i=1..m} d(w_i) = n$*) such that the valuation of* Y *lies in* U*. The reset map* R *assigns to each point in*

$G \times U$ (for the case $vp = ?U$) or to each point in G (for the cases $vp = !Y$ and $vp = \emptyset$) for each state variable $v \in \nu(l')$ a probability measure on the state space of v at location l'.

We formalize the notion of state data communication by adding three composition rules to $|^P_A|$ called *r1data*,*r2data* and *r2data'*:

$$r1data. \frac{l_1 \overset{a,G_1,R_1,v_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2,v_2}{\longrightarrow} l_2'}{l_1|^P_A|l_2 \overset{a,G_1|G_2,R_1\times R_2,v_1|v_2}{\longrightarrow} l_1'|^P_A|l_2'} (a \in A, v_1|v_2 \neq \perp).$$

Here, $l_1 \overset{a,G_1,R_1,v_1}{\longrightarrow} l_1'$ means $(l_1, a, l_1', G_1, R_1, v_1) \in \mathcal{A}_X$ with $v_1 \neq \emptyset$. Active transitions with value passing identifier equal to $\emptyset$ will be denoted as before (like $l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1'$ for example). Furthermore, $v_1|v_2$ is defined as: $v_1|v_2 := !Y$ if $v_1 = !Y$ and $v_2 := ?U$ and $\dim(U)=\dim(Y)$ or if $v_2 = !Y$ and $v_1 := ?U$ and $\dim(U)=\dim(Y)$; $v_1|v_2 := ?(U_1 \cap U_2)$ if $v_1 = ?U_1$ and $v_2 = ?U_2$ and $\dim(U_1)=\dim(U_2)$; $v_1|v_2 := \perp$ otherwise. Here $\perp$ means that v_1 and v_2 are not compatible.

$G_1|G_2$ is, only when $v_1|v_2 \neq \perp$, defined as follows: $G_1|G_2 := (G_1 \cap U) \times G_2$ if $v_1 = !Y$ and $v_2 = ?U$; $G_1|G_2 := G_1 \times (G_2 \cap U)$ if $v_1 = ?U$ and $v_2 = !Y$; $G_1|G_2 := G_1 \times G_2$ if $v_1 = ?U_1$ and $v_2 = ?U_2$. Here, $G \cap U$, which is abuse of notation, contains all state valuations x such that $x \in G$ and $Y(x) \in U$, where $Y(x)$ is the value of the ordered tuple Y according to valuation x.

In these definitions of $v_1|v_2$ and $G_1|G_2$ we see an interplay between the state guards G_1,G_2 and the input guards U_1,U_2: in the synchronization of an $(l_1, a, l_1', G_1, R_1, !Y)$ transition with a $(l_2, a, l_2', G_2, R_2, ?U)$ transition, U restricts the guard G_1 such that the Y-part of G_1 lies in U. This restriction can not be coded in $v_1|v_2$ (as it is done in the $?U_1$-$?U_2$-case), therefore we need to code it in the state guards.

Composition rules *r2data* and *r2data'* are defined as follows.

$$r2data. \frac{l_1 \overset{a,G_1,R_1,v_1}{\longrightarrow} l_1'}{l_1|^P_A|l_2 \overset{a,G_1\times val(l_2),R_1\times Id,v_1}{\longrightarrow} l_1'|^P_A|l_2} (a \notin A).$$

The mirror of *r2data* is then defined as:

$$r2data'. \frac{l_2 \overset{a,G_2,R_2,v_2}{\longrightarrow} l_2'}{l_1|^P_A|l_2 \overset{a,val(l_1)\times G_2,Id\times R_2,v_2}{\longrightarrow} l_1|^P_A|l_2'} (a \notin A).$$

Definition 7. *If $X = (L_X, V_X, \nu_X, W_X, \omega_X, F_X, G_X, \Sigma, \mathcal{A}_X, \mathcal{P}_X, \mathcal{S}_X)$ and $Y = (L_Y, V_Y, \nu_Y, W_Y, \omega_Y, F_Y, G_Y, \Sigma, \mathcal{A}_Y, \mathcal{P}_Y, \mathcal{S}_Y)$ are two value passing CPDPs that have the same set of events Σ and if we have $V_X \cap V_Y = W_X \cap W_Y = \emptyset$, then $X|^P_A|Y$ is defined as in Definition 3 except that besides the rules r1,r2,r2',r3,r3',r4,r4',r5,r6,r6',r7 and r7' for the operator $|^P_A|$ we also have the rules r1data,r2data and r2data'.*

6 Value Passing CPDP and CPDP-to-PDP Conversion: An ATM Example

6.1 ATM Example of Value Passing CPDP

In Figure 7 we see five value-passing CPDPs: *CurrentGoal*, *AudioAlert*, $Memory$, $HMI{-}PF$ and $TaskPerformance$. Together, these five components form a part of a system that models the behavior of a pilot which is controlling a flying aircraft. This pilot is called the pilot-flying. (Normally, there is also another pilot in the cockpit called the pilot-not-flying who is not directly controlling the aircraft). This example comes from Chapter 16 of this book, where it is modelled as a Dynamically Coloured Petri Net (DCPN). In this section we model an abstract version of this system as a value-passing CPDP. We first give a global description of the system. After that we give a more detailed description of each CPDP component.

There are seven distinct goals defined for the pilot-flying, C1 till C7. Which goal should be achieved by the pilot at which time depends on the situation. If at some time t_1, the pilot is working on goal C1 (which is: collision avoidance) then CPDP *CurrentGoal* is in location l_1 with $k = 1$ (the value of k equals the number of the goal) and CPDP $TaskPerformance$ is in the top location (meaning that the pilot is performing tasks for some goal while the bottom loction means that the pilot is not working an a goal). If the pilot is working on goal C2 (which is: emergency actions), then $k = 2$ and then the value q denotes which specific emergency action is executed (if $k \neq 2$ then q, which is not relevant then, equals zero). The pilot can switch to another goal in two ways:

1. He achieved a goal and is ready for a new goal. He 'looks' at the memory-unit whether there is another goal that needs to be achieved. In that case the pilot starts working on the goal in the memory-unit with the highest priority (C_1 has priority over C_2, C_2 over C_3 etc.), unless he sees on the display of $HMI{-}PF$, which is a failure indicator device, that certain aircraft-systems are not working properly. In the latter case the pilot should switch to goal C2 (emergency action).
2. The pilot is working on a goal, while CPDP *AudioAlert*, which is a communication device that can communicate alert messages, sends an *alert*-message. This message contains a value (communicated via value-passing communication) which denotes the interrupt-goal. CPDP *CurrentGoal* receives this message and if the interrupt goal has higher priority than the goal that is worked on, the pilot switches to the interrupt-goal. If the interrupt-goal has lower priority, the goal is stored into the memory-unit.

We now briefly say how the interactions between the five components are modelled: CPDP *CurrentGoal* reads the memory and the failure-indicators via value-passing-synchronization on events *getmem* and $getHMI$ respectively (see Figure 7). *CurrentGoal* receives alert-messages via value-passing-synchronization on event *alert*. $TaskPerformance$ sends the active signal

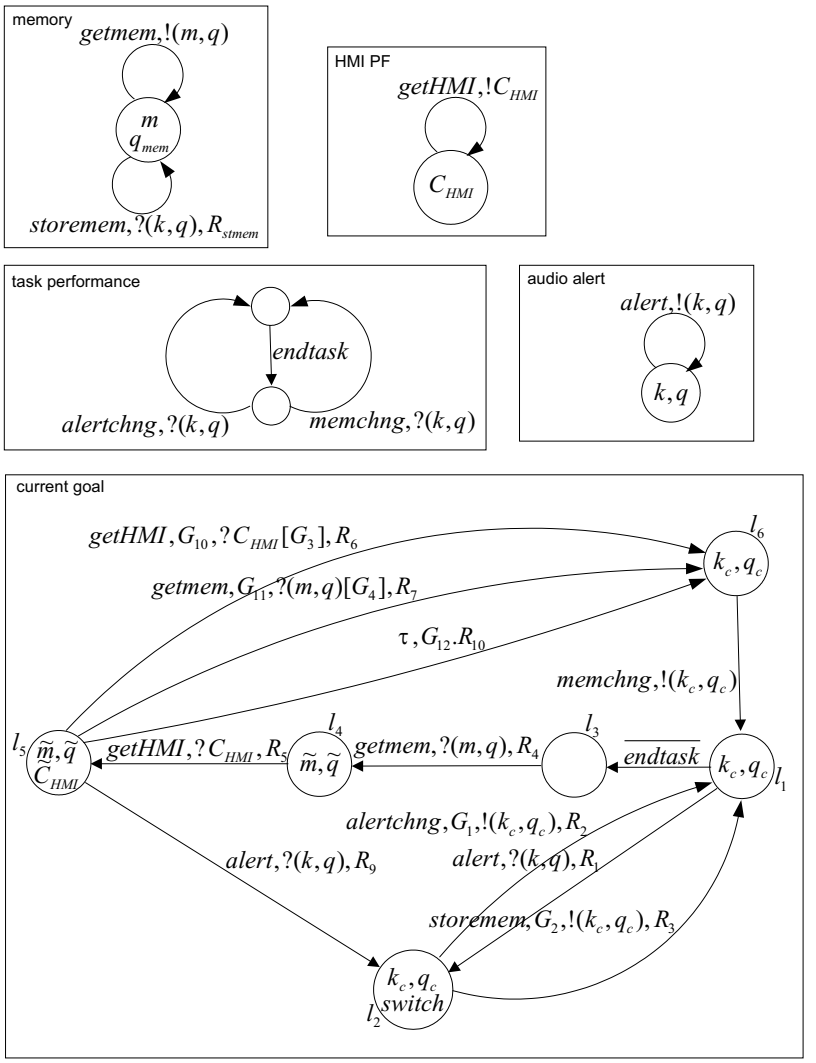

Fig. 7. CPDP pilot flying model

endtask as soon as the pilot finished the last task of the goal he was working on, this signal is received by *CurrentGoal* via a passive $\overline{endtask}$-transition. *CurrentGoal* stores a value in the memory-unit *Memory* via a value-passing-synchronization on event *storemem*. Finally, *CurrentGoal* communicates to $TaskPerformance$ that a new goal is started because of an alert-message or because a new goal was retrieved from the memory, via value-passing-synchronization on events *alertchng* and *memchng* respectively.

The five CPDPs are interconnected via composition operators of the $|_A^P|$ type as

$$(((CurrentGoal|_{A_1}|AudioAlert)|_{A_2}|Memory)$$
$$|_{A_3}|TaskPerformance)|_{A_4}|HMI{-}PF, \tag{4}$$

with $A_1 := \{alert\}$, $A_2 := \{getmem, storemem\}$, $A_3 := \{alertchng, memchng\}$ and $A_4 = \{getHMI\}$. We now describe each of the five CPDPs in more detail.

CPDP *HMI-PF* has one location with one variable named C_{HMI}. The value of this variable indicates whether there is a failure in one of the five systems (indicated by *HMI-PF*). C_{HMI} consists of five components C^i_{HMI} $(i = 1, 2, 3, 4, 5)$ which all have either value $true$ or $false$ (with $true$ indicating a failure for the corresponding system). There is only one transition, which is an unguarded active transition from the only location to itself with label $getHMI$ and with output C_{HMI}. This transition is used only to send the state information to the component *CurrentGoal*, therefore the reset map of this transition does not change the state C_{HMI}. Note that for the CPDPs in this ATM-example, we do not define output variables. We assume that for every state variable used in active transitions we have an output variable copy defined.

CPDP *AudioAlert* has one location with two variables named k and q. $k \in \{1, 2, 3, 4, 5, 6, 7\}$ and $q \in \{1, 2, 3, 4, 5, 6\}$. These values represent the interrupt goal (and failure in case $k = 2$). There is one active transition with label *alert* and with outputs k and q. This transition should normally be guarded (where the guard is satisfied as soon as an alert signal should be sent), but at the abstraction level of our model we do not model this. Also the reset map of this transition is not specified here.

CPDP *Memory* has one location with two variables named m and q_{mem}. m is a variable with seven components (m_1 till m_7 for the goals C1 till C7) which can have value ON and OFF. (In the DCPN model of this system there is also the value $LATER$ for m_4 and m_5 which we do not consider in the CPDP). q_{mem} is a variable with six components (for the six failures) taking values in $\{0, 1\}$. There are two active transitions. The unguarded transition with label *getmem* and output m and q_{mem} is used to send information to *CurrentGoal*, therefore the reset map leaves the state unaltered. The unguarded transition with label *storemem* and input k and q is used by *CurrentGoal* to change the memory state. (Note that we write $?(k, q)$ to denote inputs of the combined state-space of k and q which is $?\mathbb{R}^2$ because $k, q \in \mathbb{R}$). The reset map R_{stmem} of this transition changes m_k (with k the received input) to ON and changes q^q_{mem} (with q the received input) to 1.

CPDP *TaskPerformance* has two locations, $Idle$ and $Busy$, both without variables. When the system switches from $Busy$ to $Idle$, the active transition with label *endtask* is executed. The system can switch from $Idle$ to $Busy$ via two transitions: 1. Via the active input transition with label *alertchng* and inputs k and q. This happens when *CurrentGoal* executes an active output tran-

sition with label *alertchng* due to having received a signal from *AudioAlert*. (Normally *TaskPerformance* should use the information from the inputs k and q via the reset map of the transition, but we do not model that at our level of abstraction). 2. Via the active input transition with label *memchng* and inputs k and q. This happens when *CurrentGoal* executes an active output transition with label *memchng* due to the situation where the pilot is idling and a new goal is retrieved by *CurrentGoal* from the memory.

CPDP *CurrentGoal* is the only CPDP that we have modelled in detail. *CurrentGoal* has six locations, named l_1 till l_6. We will now describe each location:

- Location l_1 has two variables named k_c and q_c. The process is in this location when one of the goals is being achieved (i.e. *TaskPerformance* is in location *Busy*) and the values of k_c and q_c represent the current goal and (in case $k_c = 2$) current failure. There are two outgoing transitions: 1. An unguarded active input transition to l_2 labelled *alert* with inputs k and q, synchronizing on an *alert* signal from *AudioAlert*, with reset map

$$R_1 := \begin{cases} k_c := k, q_c := q, switch := true & \text{if k} < \text{k}_\text{c} \\ k_c := k_c, q_c := q_c . switch := false & \text{else.} \end{cases}$$

  2. A passive transition to l_3 labelled $\overline{endtask}$, synchronizing on an *endtask* signal from *TaskPerformance*.
- The process is in location l_2 when (1) after having received the *alert* signal the current goal needs to be changed (according to the *alert* signal) or when (2) the interrupt goal (from the *alert* signal) needs to be stored in memory. (1) is the case when $switch = true$, (2) is the case when $switch = false$. Therefore, $G_1 := \{(k_c, q_c, switch) | switch = true\}$, $G_2 := \{(k_c, q_c, switch) | switch = false\}$, with G_1 the guard of the active output transition labelled *alertchng* with outputs k_c and q_c and reset map R_2 and with G_2 the guard of the active output transition labelled *storemem* with outputs k_c and q_c and reset map R_3. R_2 and R_3 are the same and do the following reset: $k_c := k_c$, $q_c := q_c$. Note that, under maximal progress, the process jumps immediately to location l_1 as soon as it arrives in location l_2, causing also a synchronizing transition in either *TaskPerformance* (with label *alertchng*) or *Memory* (with label *storemem*).
- The process arrives in location l_3 after the *endtask* signal. Then the pilot should check the memory whether there are other goals that need to be achieved. With the unguarded active input transition with label *getmem* and inputs m and q and reset map R_4, the process jumps to location l_4 while retrieving the memory state (m, q). The reset map R_4 stores this (m, q) in $(\tilde{m}, \tilde{q})$.
- Before executing a goal from the memory, the pilot should first check *HMI-PF* to see whether there are indications for failing devices. This happens in the transition to l_5 on the label $getHMI$ while retrieving the *HMI-PF*

state C_{HMI}. The reset R_5 stores C_{HMI} together with $\tilde{m}$ and $\tilde{q}$ in the state of l_5.

- From location l_5 there is an active transition to l_6 with label τ and guard $G_{12} := \{(\tilde{m}, \tilde{q}, \tilde{C}_{HMI}) |\ \tilde{C}^i_{HMI} = true$ for some $i = 1, 2, 3, 4, 5$ or $\tilde{m}^i = ON$ for some $i < 7\}$. Under maximal progress, this τ-transition is taken immediately after arriving in l_5 when the *Memory* and *HMI-PF* states give reason to work on a new goal. The reset map R_{10} resets $k_c := 2, q_c := r$ if $S := \{i | i \leq 5, \tilde{C}^i_{HMI} = true\} \neq \emptyset$, where r is randomly chosen from the set S, otherwise R_{10} resets $k_c := \min\{i | m_i = ON\}, q_c := 0$. If the guard G_{12} is not satisfied in l_5, then this means that the pilot should wait until an *alert* signal is received or until either the *Memory* state or the *HMI-PF* state changes such that the pilot should work on a new goal. On an *alert* signal from *AudioAlert* the transition to l_2 is taken where R_9 is equal to R_1. The active input transition to l_6 labelled *getmem* waits till the *Memory* state has changed such that the input-guard G_4 is satisfied, where $G_4 := \{(m, q) | m^i = ON$ for some $2 \neq i < 7\}$. The reset map R_7 resets $k_c := \min\{i | m_i = ON\}, q_c := 0$. The active input transition to l_6 labelled *getHMI* waits till the *HMI-PF* state has changed such that the input-guard G_3 is satisfied, where $G_3 := \{C_{HMI} | C^i_{HMI} = true$ for some $i = 1, 2, 3, 4, 5\}$. The reset map R_6 resets $k_c := 2, q_c := r$ with r randomly chosen from $S := \{i | i \leq 5, \tilde{C}^i_{HMI} = true\} \neq \emptyset$.
- If the process arrives in location l_6, then this means that the state of l_6 represents the goal that should immediately be worked on by the pilot. Therefore, the unguarded active transition to l_1 labelled *memchng* is taken immediately (under maximal progress). The outputs k_c and q_c are accepted by the *memchng* transition in *TaskPerformance*. The reset map of the output *memchng* transition copies the state of l_6 to the state of l_1.

6.2 Examples of Value-Passing-CPDP to PDP Conversion

We follow the algorithm from Section 4.1 to check whether the CPDP ATM-example of Section 6, which has no spontaneous transitions, can be converted to a PDP.

Example 3 (ATM). We assume that the system modelled by (4) is closed (i.e. no more components will be connected). This means that we remove the passive transitions in the composite CPDP (which are some $\overline{endtask}$ transitions). It can be seen that the composite CPDP does not have active input-transitions. We assume that time will elapse in the locations of *AudioAlert* and $TaskPerformance$. Both may have (different) extra dynamics of the form $\dot{x} = f(x)$, then the guards of transitions *alert* and *endtask* depend on x. We assume that the transitions *alert*, *alertchng* and *memchng* are stable. Note that location l_1 is unguarded, that locations l_2,l_3,l_4 and l_6 are guarded and that location l_5 has both an unguarded and a guarded state space.

First we look at T^1_s: the stable parts of the transitions that represent hybrid jumps of multiplicity one. For this example we have

$$T_s^1 = \{storemem, alertchng, memchng, getHMI_{s,45}\},$$

where these names correspond to the transitions with the same label in Figure 7: *storemem* represents the transition from l_2 to l_1 synchronized with the transition with the same label in component *memory*. $getHMI_{s,45}$ corresponds to the stable part, which is the part that does not jump into guard G_{12}, of the transition between l_4 and l_5 synchronizing with the transition in *HMI-PF*, etc. Because R_5 makes a copy of C_{HMI},m and q, we get that the guard of $getHMI_{s,45}$ equals $val(l_4)\backslash G_{12}$ and the guard of $getHMI_{u,45}$, the unstable part, equals G_{12}. Furthermore, we have for this example

$$T_u^1 = \{alert_{12}, alert_{52}, getmem_{34}, getmem_{56}, getHMI_{u,45}, getHMI_{56},$$

$$endtask\}, \quad T_s^2 = \{alertchng \circ alert_{12}, alertchng \circ alert_{52}, storemem \circ alert_{12},$$

$$storemem \circ alert_{52}, memchng \circ \tau, memchng \circ getHMI, memchng \circ getmem,$$

$$getHMI_s \circ getmem\},$$

where $getHMI_s \circ getmem$ denotes the transition that represents the hybrid jump of multiplicity two that consists of *getmem* from l_3 to l_4 followed directly by the stable part of *getHMI* from l_4 to l_5, etc. Then,

$$T_u^2 = \{getmem \circ endtask, getHMI_u \circ getmem, \tau \circ getHMI\},$$

$$T_s^3 = \{memchng \circ \tau \circ getHMI_u, getHMI_s \circ getmem \circ endtask\},$$

$$T_u^3 = \{getHMI_u \circ getmem \circ endtask, \tau \circ getHMI_u \circ getmem\},$$

$$T_s^4 = \{memchng \circ \tau \circ getHMI_u \circ getmem\},$$

$$T_u^4 = \{\tau \circ getHMI_u \circ getmem \circ endtask\}.$$

$$T_s^5 = \{memchng \circ \tau \circ getHMI_u \circ getmem \circ endtask\},$$

$$T_u^5 = \emptyset.$$

We see, when X denotes the composite CPDP, that X^5 (i.e. the CPDP that has active transitions $(\cup_{i=1}^5 T_s^i) \cup T_u^5$) has no unstable transitions. This means that X^5 can directly be converted to a PDP, which then is the corresponding PDP of X.

To prove that the composite CPDP of this ATM example can be converted to a PDP, it would also have been enough to show that the CPDP does not have cycles such that the locations of the cycle all have guarded parts. It is clear that a cycle in component *Current goal* should include location l_1, which is an unguarded location. It can easily be seen that in the composite CPDP the two (product)locations that contain l_1 are both unguarded and that any cycle in the composite CPDP should contain one of these two locations. Therefore this composite CPDP does not have transitions with multiplicity infinity and should therefore be convertable to a PDP. (However, if we want to specify this PDP, we still have to do the algorithm or something similar).

Because the algorithm terminates on the ATM-example above, we know that the ATM-example has a PDP behavior. However, it is possible that the algorithm does not terminate, while the CPDP does exhibit a PDP behavior. We now give an example of this.

Example 4. Let CPDP X have one location, l_1. The state-space of l_1 is $[0,1]$, the continuous dynamics of l_1 is the clock dynamics $\dot{x} = 1$. From l_1 to l_1 there is one active transition with guard G and reset map R. $G = [\frac{1}{2}, 1]$. For $x \in G$, $R(\{0\}, x) = \frac{1}{2}$ and $R(A, x) = |A \cap [\frac{1}{2}, 1]|$ for $A \in \mathcal{B}([0,1]\backslash\{0\})$. This means that from an x in G, the reset map jumps to 0 with probability $\frac{1}{2}$ and jumps uniformly into $[\frac{1}{2}, 1]$ with probability $\frac{1}{2}$. It can easily be seen that for X we have that $T_u^n \neq \emptyset$ for all $n \in \mathbb{N}$. This means that the algorithm explained above does not terminate for this example. Still, according to Theorem 2, X expresses a PDP behavior, because for $x \in G$, $R([0,1], x) = \lim_{n\to\infty} R_{tot,s}^n([0,1], x) = \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} + \cdots = 1$.

7 Bisimulation for CPDPs

In this section we define bisimulation relations for CPDPs. Bisimulation is an equivalence relation. The idea of bisimulation is that two CPDPs are bisimulation-equivalent if for an external agent the CPDPs cannot be distinguished from each other. We assume here that an external agent cannot see the state-value of a CPDP but it does see the output-value of a CPDP and it does also see the events (including possible value passing information) of active transitions. We assume that the behavior of the external agent can be modelled as another CPDP. Thus, if CPDPs X_1 and X_2 are bisimilar (i.e. bisimulation-equivalent), then $X_1|_A^P|Y$ and $X_2|_A^P|Y$ behave externally equivalently for each external-agent-CPDP Y and each operator of the form $|_A^P|$. *External equivalent behavior* will be defined later in this section, but for the intuitive understanding, we will already give two examples here.

1. Suppose the initial states of CPDPs X_1, X_2 are given. If then, for some CPDP Y (with some initial state) and some $|_A^P|$, the probability that the output-value of $X_1|_A^P|Y$ equals $\hat{w}$ at time $\hat{t}$, is different from the probability that the output-value of $X_2|_A^P|Y$ equals $\hat{w}$ at time $\hat{t}$, then X_1 and X_2 are not bisimilar.

2. As an example of two bisimilar CPDPs, we compare CPDP X from Figure 4 to CPDP $\tilde{X}$ from Figure 8. We let $\tilde{\lambda}$, $\tilde{\mu}$, all $\tilde{G}_i$ and all $\tilde{R}_i$ be copies of λ,μ,G_i and R_i from Figure 4, i.e. $\tilde{\lambda}$, $\tilde{\mu}$, $\tilde{G}_i$ and the $\tilde{x}$-resets of $\tilde{R}_i$ do not depend on $\bar{x}$. The $\bar{x}$ resets of $\tilde{R}_i$ are not relevant here and may therefore be chosen arbitrarily (like $\bar{x} := 0$ for each $\tilde{R}_i$). Thus, we get $\tilde{\lambda}(\tilde{x}, \bar{x}) = \lambda(\tilde{x})$, $\tilde{G}_i = \{(\tilde{x}, \bar{x}) | \tilde{x} \in G_i\}$, etc. Then, the only difference between X and $\tilde{X}$, if we regard $\tilde{x}$ as a copy of x, is that the locations of $\tilde{X}$ have another state variable $\bar{x}$ (evolving along vectorfields $\bar{f}_1$ and $\bar{f}_2$). But this extra variable $\bar{x}$ does not influence the output y, which only depends on x (or $\tilde{x}$), and it also

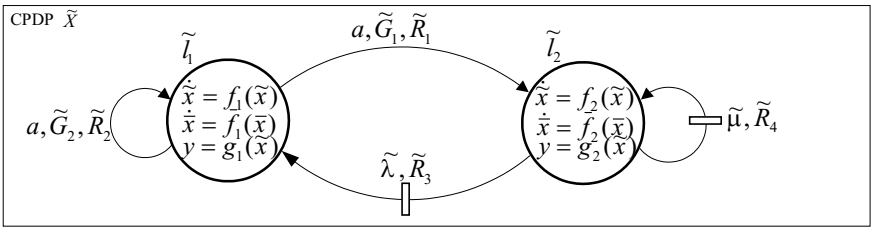

Fig. 8. CPDP $\tilde{X}$ (bisimulation equivalent to CPDP X of Figure 4)

does not influence hybrid jumps because it does not influence the guards of the transitions, the Poisson processes and the resets of x (or $\tilde{x}$). It is intuitively clear then that CPDPs X and $\tilde{X}$ cannot be distinguished by an external agent. After the formal definition of bisimulation for CPDPs, we will show that X and $\tilde{X}$ are indeed bisimilar.

X can be seen as a state reduced equivalent of $\tilde{X}$ because the state space of X is smaller (i.e. the variable $\bar{x}$ is not present in X). More formally, we could say that we have state reduction because each state x of X represents a whole set of states $\{(\tilde{x}, \bar{x}) | \tilde{x} = x\}$ of $\tilde{X}$ (i.e. the state valuation $(x = 1)$ of X for example, represents the set of state valuations $\{(\tilde{x} = 1, \bar{x} = r) | r \in \mathbb{R}\}$ of $\tilde{X}$). State valuation $(\tilde{x} = 1, \bar{x} = 0)$ is for example equivalent to state valuation $(\tilde{x} = 1, \bar{x} = 1)$ because the external behavior of $\tilde{X}$ that starts/continues from $(\tilde{x} = 1, \bar{x} = 0)$ is the same as the external behavior of $\tilde{X}$ that starts/continues from $(\tilde{x} = 1, \bar{x} = 1)$. We could say therefore that $\{(\tilde{x} = 0, \bar{x} = r) | r \in \mathbb{R}\}$ forms an equivalence class of states. In the formal definition of bisimulation for CPDPs, we will see that we can indeed use this concept of equivalence classes of states. Before we do that, we need to introduce the technical concepts of *induced equivalence relation*, *measurable relation* and *equivalent (probability) measure*.

We define the equivalence relation on X that is induced by a relation $\mathcal{R} \subset X \times Y$ with the property that $\pi_1(\mathcal{R}) = X$ and $\pi_2(\mathcal{R}) = Y$, where $\pi_i(\mathcal{R})$ denotes the projection of $\mathcal{R}$ on the i-th component, as the transitive closure of $\{(x, x') | \exists y$ s.t. $(x, y) \in \mathcal{R}$ and $(x', y) \in \mathcal{R}\}$. We write $X/_{\mathcal{R}}$ and $Y/_{\mathcal{R}}$ for the sets of equivalence classes of X and Y induced by $\mathcal{R}$. We denote the equivalence class of $x \in X$ by $[x]$. We will now define the notions of *measurable relation* and of *equivalent measure*.

Definition 8. *Let $(X, \mathcal{X})$ and $(Y, \mathcal{Y})$ be Borel spaces and let $\mathcal{R} \subset X \times Y$ be a relation such that $\pi_1(\mathcal{R}) = X$ and $\pi_2(\mathcal{R}) = Y$. Let $\mathcal{X}^*$ be the collection of all $\mathcal{R}$-saturated Borel sets of X, i.e. all $B \in \mathcal{X}$ such that any equivalence class of X is either totally contained or totally not contained in B. It can be checked that $\mathcal{X}^*$ is a σ-algebra. Let*

$$\mathcal{X}^*/_{\mathcal{R}} = \{[A] | A \in \mathcal{X}^*\},$$

where $[A] := \{[a]|a \in A\}$. *Then* $(X/_{\mathcal{R}}, \mathcal{X}^*/_{\mathcal{R}})$, *which is a measurable space, is called the* quotient space *of* X *with respect to* $\mathcal{R}$. *A unique bijective mapping* $f : X/_{\mathcal{R}} \to Y/_{\mathcal{R}}$ *exists, such that* $f([x]) = [y]$ *if* $(x, y) \in \mathcal{R}$. *We say that the relation* $\mathcal{R}$ *is* measurable *if for all* $A \in \mathcal{X}^*/_{\mathcal{R}}$ *we have* $f(A) \in \mathcal{Y}^*/_{\mathcal{R}}$ *and vice versa.*

If a relation on $X \times Y$ is measurable, then the quotient spaces of X and Y are homeomorphic (under bijection f from Definition 8). We could say therefore that under a measurable relation X and Y have a shared quotient space. In the field of descriptive set theory, a relation $\mathcal{R} \subset X \times Y$ is called measurable if $\mathcal{R} \in \mathcal{B}(X \times Y)$ (i.e. $\mathcal{R}$ is a Borel set of the space $X \times Y$). This definition does not coincide with our definition of measurable relation. In fact, many interesting measurable relations are not Borel sets of the product space $X \times Y$.

Definition 9. *Suppose we have measures* P_X *and* P_Y *on Borel spaces* $(X, \mathcal{X})$ *and* $(Y, \mathcal{Y})$ *respectively. Suppose that we have a measurable relation* $\mathcal{R} \subset X \times Y$. *The measures* P_X *and* P_Y *are called equivalent with respect to* $\mathcal{R}$ *if we have* $P_X(f_X^{-1}(A)) = P_Y(f_Y^{-1}(f(A)))$ *for all* $A \in \mathcal{X}^*/_{\mathcal{R}}$ *(with* f *as in Definition 8 and with* f_X *and* f_Y *the mappings that map* X *and* Y *to* $X/_{\mathcal{R}}$ *and* $Y/_{\mathcal{R}}$ *respectively).*

As an example, we show that relation $\mathcal{R} = \{(x, (\tilde{x}, \bar{x}))|x = \tilde{x}\}$ on $val(X) \times val(\tilde{X})$, where $val(X)$ and $val(\tilde{X})$ denote the state spaces of CPDPs X and $\tilde{X}$ of Figures 4 and 8, is a measurable relation and that the reset maps $R_i(x)$ and $\tilde{R}_i(\tilde{x}, \bar{x})$ are equivalent measures under this relation if $f([x]) = ([\tilde{x}, \bar{x}])$: the induced equivalence relation of $\mathcal{R}$ on X equals $\{\{x\}|x \in val(X)\}$, i.e. each single valuation forms an equivalence class of X. The induced equivalence relation of $\mathcal{R}$ on $\tilde{X}$ equals $\{\{(\tilde{x} = q, \bar{x} = r)|r \in \mathbb{R}\}|q \in \mathbb{R}\}$. The saturated Borel sets of X are all Borel sets of X, the saturated Borel sets $\tilde{X}$ are all sets of the form $B \times \mathbb{R}$ with B a Borel set for the state $\tilde{x}$ (i.e. a Borel set of $\mathbb{R}$). The bijective mapping f from Definition 8 maps each saturated Borel set B of X to the saturated Borel set $B \times \mathbb{R}$ of Y, from which follows, according to Definition 8, that $\mathcal{R}$ is measurable.

If states x and $(\tilde{x}, \bar{x})$ are equivalent (i.e. $f([x]) = [(\tilde{x}, \bar{x})]$), then the measures $R_i(\cdot, x)$ and $\tilde{R}_i(\cdot, (\tilde{x}, \bar{x}))$ are equivalent because R_i and $\tilde{R}_i$ are defined such that for each (saturated borel set of X) $B \in \mathcal{B}(\mathbb{R})$ we have $R_i(B, x) = \tilde{R}_i(B \times \mathbb{R}, (\tilde{x}, \bar{x}))$.

In order to define bisimulation for CPDPs we also need to introduce the notions of *combined reset map* and *combined jump rate function*: we consider CPDP (without value passing) $X = (L, V, W, v, w, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$, with hybrid state space $E = E_s \cup E_u$, together with scheduler S. We define R, which we call the combined reset map, as follows. R assigns to each triplet (l, x, a) with $(l, x) \in E_u$ and with $a \in \Sigma$ such that $l \xrightarrow{a}$ (i.e. there exists an active transition labelled a leaving l), a measure on E. This measure $R(l, x, a)$ is for any l' and any Borel set $A \subset val(l')$ defined as:

$$R(l,x,a)(l',A) = \sum_{\alpha \in \mathcal{A}_{l,a,l'}} S(l,x)(\alpha) R_\alpha(A,x),$$

where $\mathcal{A}_{l,a,l'}$ denotes the set of active transitions from l to l' with label a and (l',A) denotes the set $\{(l',x)|x \in A\}$. (This measure is uniquely extended to all Borel sets of E). Now, for $A \in \mathcal{B}(E)$, $R(l,x,a)(A)$ equals the probability of jumping into A via an active transition with label a given that the jump takes place at (l,x).

Furthermore, R assigns to each triplet $(l,x,\bar{a})$ with $(l,x) \in E$ and with $\bar{a} \in \bar{\Sigma}$ such that $l \xrightarrow{\bar{a}}$, a measure on E, which for any l' and any Borel set $A \subset val(l')$ is defined as:

$$R(l,x,\bar{a})(l'_1,A) = \sum_{\alpha \in \mathcal{P}_{l,\bar{a},l'}} S(l,x)(\alpha) R_\alpha(A,x).$$

(This measure is uniquely extended to all Borel sets of E). Now, $R(l,x,\bar{a})(A)$, with $A \in \mathcal{B}(E)$, equals the probability of jumping into A if a passive transition with label $\bar{a}$ takes place at (l,x).

We define the combined jump rate function λ for CPDP X as

$$\lambda(l,x) = \sum_{\alpha \in \mathcal{S}_{l\rightarrow}} \lambda_\alpha(l,x),$$

with $(l,x) \in E$.

Finally, for spontaneous jumps, R assigns to each $(l,x) \in E$ such that $\lambda(l,x) \neq 0$, a probability measure on E, which for any l' and any Borel set $A \subset val(l')$ is defined as:

$$R(l,x)(l'_1,A) = \sum_{\alpha \in \mathcal{S}_{l\rightarrow l'}} \frac{\lambda_\alpha(l,x)}{\lambda(l,x)} R_\alpha(A,x).$$

(This measure is uniquely extended to all Borel sets of E). Now we are ready to give the definition of bisimulation for CPDPs.

Definition 10. *Suppose we have CPDPs* $X = (L_X, V_X, W, v_X, w_X, F_X, G_X, \Sigma, A_X, P_X, S_X)$ *and* $Y = (L_Y, V_Y, W, v_Y, w_Y, F_Y, G_Y, \Sigma, A_Y, P_Y, S_Y)$ *with shared* W *and* Σ *and with schedulers* S_X *and* S_Y. *A measurable relation* $\mathcal{R} \subset val(X) \times val(Y)$ *is a bisimulation if* $((l_1,x),(l_2,y)) \in \mathcal{R}$ *implies that*

1. $\omega_X(l_1) = \omega_Y(l_2)$, *for all* $w \in \omega_X(l_1)$ *we have* $G_X(l_1,x,w) = G_Y(l_2,y,w)$, $\lambda(l_1,x) = \lambda(l_2,y)$ *(with* λ *the combined jump rate function defined on both* $val(X)$ *and* $val(Y)$*).*
2. $(\phi_{l_1}(t,x), \phi_{l_2}(t,y)) \in \mathcal{R}$ *(with* $\phi_l(t,z)$ *the state at time* t *when the state equals* z *at time zero).*
3. *If* $\lambda(l_1,x) = \lambda(l_2,y) \neq 0$, *then* $R(l_1,x)$ *and* $R(l_2,y)$ *are equivalent probability measures with respect to* $\mathcal{R}$.

4. *For any* $\bar{a} \in \bar{\Sigma}$ *we have that either both* $l_1 \not\xrightarrow{\bar{a}}$ *and* $l_2 \not\xrightarrow{\bar{a}}$ *or else* $R(l_1, x, \bar{a})$ *and* $R(l_2, y, \bar{a})$ *are equivalent probability measures.*
5. *For any* $a \in \Sigma$ *we have that either both* $l_1 \not\xrightarrow{a}$ *and* $l_2 \not\xrightarrow{a}$ *or else* $R(l_1, x, a)$ *and* $R(l_2, y, a)$ *are equivalent measures.*

X with initial state (l_1, x) and Y with initial state (l_2, y) are bisimilar if $((l_1, x), (l_2, y))$ is contained in some bisimulation.

Definition 10 formalizes what we mean by equivalent external behavior. It can now be seen that, according to Definition 10, CPDP X (from Figure 4) with initial state (l_x, x) (for some l_x and some $x \in val(l_x)$) together with some scheduler S_X, and CPDP $\tilde{X}$ (from Figure 8) with initial state $(l_{\tilde{x}}, (\tilde{x}, \bar{x}))$ (with $l_{\tilde{x}} = l_x$ and $\tilde{x} = x$ and $\bar{x} \in \mathbb{R}$) together with scheduler $S_{\tilde{X}}(\tilde{l}, (\tilde{x} = q, \bar{x} = r))(\tilde{\alpha}) := S_X(l, x = q)(\alpha)$ (where $\tilde{\alpha}$ is the transition of $\tilde{X}$ that corresponds according to Figures 4 and 8 to transition α of X) are bisimilar under the relation $\mathcal{R} = \{(x, (\tilde{x}, \bar{x})) | x = \tilde{x}\}$ on $val(X) \times val(\tilde{X})$ (which was already shown to be a measurable relation).

We now state a theorem which justifies our notion of bisimulation when it concerns the stochastic behavior. It says that if two closed CPDPs are bisimilar, then the stochastic processes that model the output evolution of the CPDPs are equivalent (in the sense of indistinguishability).

Theorem 3. *The stochastic processes of the outputs of two bisimilar closed CPDPs (with their schedulers), whose quotient spaces are Borel spaces, can be realized such that they are indistinguishable.*

Proof. The proof can be found in [15]. There, *invariants* are used instead of guards. It can be seen that the proof is still valid if the invariant of a location is defined as the unguarded state space of that location.

It can easily be seen that if two non-closed CPDPs are bisimilar, then if we close both CPDPs (i.e. if we remove all passive transitions), then the closed CPDPs are still bisimilar and, by Theorem 3, the stochastic processes that model the output evolution of the CPDPs are equivalent.

We now state a theorem which justifies our notion of bisimulation when it concerns the interaction behavior. It says that two bisimilar CPDPs interact in an equivalent way (with any other CPDP) by stating that substituting a CPDP-component (in a composition context with multiple components) by another, but bisimilar, component, results in a composite CPDP that is bisimilar to the original composite CPDP. Checking bisimilarity between two composite CPDPs can only be done if both composite CPDPs have their own schedulers. Therefore we first have to investigate how a scheduler of a composite CPDP can be composed from the schedulers of the components.

It appears that the schedulers of the components do not contain enough information to define the scheduler of the composite CPDP. We illustrate this with Figure 9, where we see two CPDPs, X and Y, with schedulers S_X and

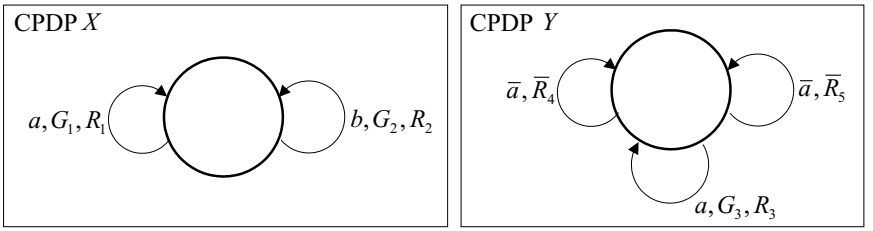

Fig. 9. Example concerning internal/external scheduling

S_Y. Suppose we connect X and Y via composition operator $|^{\bar{\Sigma}}_{\emptyset}|$. If $x \in G_1$ and $x \notin G_2$ and $y \notin G_3$, then the scheduler S of $X|^{\bar{\Sigma}}_{\emptyset}|Y$ is at (x, y) determined because (a, G_1, R_1) is the only transition that is enabled at (x, y), therefore the scheduler has to choose this transition. However, this a-transition will trigger one of the two $\bar{a}$-transitions of Y. Thus, the scheduler still has to choose between the transitions $(a, G_1 \times val(Y), R_1 \times \bar{R}_4)$ (i.e. the synchronization of (a, G_1, R_1) with $(\bar{a}, \bar{R}_4)$) and $(a, G_1 \times val(Y), R_1 \times \bar{R}_5)$. Here we should respect S_Y which is defined to make a choice between the two passive transitions. Thus we get,

$$S(x, y)(a, G_1 \times val(Y), R_1 \times \bar{R}_i) = S_Y(y, \bar{a})(\bar{a}, \bar{R}_i), \quad i \in \{4, 5\}.$$

If $x \notin G_1$ and $x \in G_2$ and $y \in G_3$, then at state (x, y), two active transitions of $X|^{\bar{\Sigma}}_{\emptyset}|Y$ are enabled: $(b, G_2 \times val(Y), R_2 \times Id)$ and $(a, val(X) \times G_3, Id \times R_3)$. S_X and S_Y give no information how to choose between the b-transition and the a-transition. We call this case a case of *external scheduling* (i.e. the choice cannot be made by the internal schedulers, the schedulers of the individual components). Thus, besides the internal schedulers S_X and S_Y, we need a strategy for external scheduling. We define this as follows.

Definition 11. *ESS is an external scheduling strategy for* $X|^{P}_{A}|Y$ *with internal schedulers* S_X *and* S_Y *if ESS assigns to each state* (x, y) *a mapping from the set of event pairs* EP *to* $[0, 1]$*, where*

$$EP := \{[\alpha, \beta] | \alpha = \beta \in \Sigma, \alpha \in \Sigma \wedge \beta = *, \alpha = * \wedge \beta \in \Sigma,$$

$$\alpha \in \Sigma \wedge \beta = \bar{\alpha}, \alpha = \bar{\beta} \wedge \beta \in \Sigma, \alpha = \beta \in \bar{\Sigma}, \alpha \in \bar{\Sigma} \wedge \beta = *, \alpha = * \wedge \beta \in \bar{\Sigma}\},$$

which respects the transition structure of $X|^{P}_{A}|Y$*.*

We explain the meaning of external scheduling strategy by using the example of Figure 9: if ESS is an external scheduling strategy for $X|^{\bar{\Sigma}}_{\emptyset}|Y$ and $ESS(x, y)([a, \bar{a}]) = 1$, then the set of transitions of the form $(a, G_x \times val(Y), R_x \times \bar{R}_y)$ (with (a, G_x, R_x) an a-transition of X and $(\bar{a}, \bar{R}_y)$ a $\bar{a}$-transition of Y) at state (x, y) get probability one. The probabilities of the individual transitions of this form are determined by the internal schedulers.

If we have $ESS(x,y)([a,\bar{a}]) > 0$ with $x \notin G_1$, then ESS does not respect the transition structure, because for $x \notin G_1$ no a-transition of X can be executed, and is therefore not a valid external scheduling strategy, etc. In general, an external scheduling strategy does not have to respect the internal schedulers where it concerns the choice between active transitions (within one component) labelled with different events, but it has to respect the internal schedulers where it concerns the passive transitions and the choice between active transitions (in one component) with the same event-label. The choice to allow to ignore internal schedulers where it concerns active transitions with different event-labels, has been made because first, in some cases it is not clear what it means to respect the internal schedulers and second, this freedom does not influence the result of the bisimulation-substitution-theorem that we state after the following example about a scheduler that does respect the internal schedulers as much as possible.

Example 5. Suppose we have two CPDPs X and Y with schedulers S_X and S_Y, which we interconnect with composition operator $|_{\emptyset}^{\bar{\Sigma}}|$. A valid external scheduling strategy would be:

- For states (x,y) with $x \in val_u(X)$ (i.e. the guarded states of X) and $y \in val_s(Y)$ the choice for the active transition of X is made by S_X. (Which passive transitions synchronize depends on Y and S_Y)
- For states (x,y) with $x \in val_s(X)$ and $y \in val_u(Y)$ the choice for the active transition of Y is made by S_Y. (Which passive transitions synchronize depends on X and S_X)
- For states (x,y) with $x \in val_u(X)$ and $y \in val_u(Y)$, the choice for the active transition (of X or Y) is determined with probability half by S_X and with probability half by S_Y. (Which passive transitions synchronize depends on X,Y, S_X and S_Y).

Note that the strategy of Example 5 will not work in case $A \neq \emptyset$. Also, in general, the composition of two schedulers under an external scheduling strategy, which results in a internal schedular for the composite system (as in Example 5), is not commutative and not associative.

Theorem 4. *Suppose we have three CPDPs, X_1,X_2 and Y, with schedulers S_{X_1}, S_{X_2} and S_Y. Suppose $\mathcal{R} \subset val(X_1) \times val(X_2)$ is a bisimulation and $val(X_1)/_{\mathcal{R}}$ and $val(X_2)/_{\mathcal{R}}$ (i.e. the quotient spaces of X_1 and X_2 under $\mathcal{R}$) are Borel spaces. Then,*

$$\mathcal{R}' := \{((x_1,y),(x_2,y))|(x_1,x_2) \in \mathcal{R}, y \in val(Y)\}$$

is a bisimulation on $(val(X_1) \times val(Y)) \times (val(X_2) \times val(Y))$ for the CPDPs $X_1|_A^P|Y$ and $X_2|_A^P|Y$ with external scheduling strategies ESS_1 and ESS_2 such that $ESS_1(x_1,y) = ESS_1(x_2,y)$ if $(x_1,x_2) \in \mathcal{R}$. Furthermore, $(val(X_1) \times val(Y))/_{\mathcal{R}'}$ and $(val(X_2) \times val(Y))/_{\mathcal{R}'}$ are Borel spaces.

Proof. The proof can be found, mutatis mutandis, in [15].

With Theorem 4, we can use bisimulation as a compositional reduction technique: suppose we want to perform stochastic analysis on a (closed) composite CPDP that consists of multiple components. To reduce the state space of this complex system, we can reduce (by bisimulation) each component individually and put the reduced state component back in the composition. In this way the state of the composite CPDP will be reduced as soon as one or more of the components are state reduced. We know that the stochastic behavior of the output evolution is not changed by bisimulation, therefore we can perform the stochastic analysis on the (closed) state reduced composite CPDP.

Bisimulation for value-passing CPDPs

The definition of bisimulation can also be defined for value-passing CPDPs. We will not do that here, but we are convinced that it can be shown that with small extensions to the operation of schedulers (such that they can handle value-passing), and to the definitions of combined reset map and external scheduling strategies, the Theorems 3 and 4 also apply to the case of value-passing CPDPs. However, this result still has to be achieved.

8 Conclusions and Discussion

In this chapter we introduced the CPDP automata framework. CPDPs are automata with labelled transitions and spontaneous (stochastic) transitions. The locations of a CPDP are enriched with state and output variables. Each state variable (of a specific location) evolves according to a specified differential equation. State variables are probabilistically reset after a transition has been executed. CPDPs can interact/communicate with each other via the event-labels of the labelled transitions. For the extended framework value-passing-CPDP, event labels may even hold information about the output variables. We defined a bisimulation notion for CPDP. We proved that bisimilar CPDPs exhibit equivalent stochastic and interaction behavior. Therefore, bisimulation can be used as a compositional state reduction technique.

This means that we can take a component from a complex CPDP, find a state reduced bisimilar component and put the state reduced component back in the composition. The problem however is: how to find a state reduced bisimilar component? For certain classes of systems, like for IMC (see [8]) and for linear input/output systems (see [16]), (decidable) algorithms have been developed to find maximal (i.e. maximally state reduced) bisimulations. Since CPDPs are very general in the stochastics and the continuous dynamics, we can not expect that similar algorithms can be developed for CPDPs also. However, we can try to find subclasses of CPDPs that do allow automatic generation of maximal bisimulations. Any complex CPDP can then in

principle be state reduced by finding the components that allow automatic generation of bisimulations and replace these components with their maximal bisimilar equivalents.

Bisimulation can be seen as a compositional analysis technique, i.e. it uses the composition structure in order to make analysis easier. Other compositional analysis techniques should benefit from the composition structure in their specific ways. In our CPDP model there is a clear distinction between the different components of a complex system and it is formalized how the composite behavior is constituted from the components and from the interaction mechanisms (i.e. the composition operators) that interconnect the components. Since we have this clear and formal composition structure (including a clear operational semantics for the composition operation), we think our model might be suitable for developing compositional analysis techniques.

References

1. T. Bolognesi and E. Brinksma. Introduction to the iso specification language lotos. *Comp. Networks and ISDN Systems*, 14:25–59, 1987.
2. P. R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, University of Twente, 1997.
3. M. H. A. Davis. Piecewise Deterministic Markov Processes: a general class of non-diffusion stochastic models. *Journal Royal Statistical Soc. (B)*, 46:353–388, 1984.
4. M. H. A. Davis. *Markov Models and Optimization*. Chapman & Hall, London, 1993.
5. S.N. Strubbe et al. On control of complex stochastic hybrid systems. Technical report, Twente University, 2004. http://www.nlr.nl/public/hosted-sites/hybridge/.
6. M. H. C. Everdij and H. A. P. Blom. Petri-nets and hybrid-state markov processes in a power-hierarchy of dependability models. In *Proceedings IFAC Conference on Analysis and Design of Hybrid Systems ADHS 03*, 2003.
7. M.H.C. Everdij and H.A.P. Blom. Piecewise deterministic Markov processes represented by dynamically coloured Petri nets. *Stochastics: An International Journal of Probability and Stochastic Processes*, 77(1):1–29, February 2005.
8. H. Hermanns. *Interactive Markov Chains*, volume 2428 of *Lecture Notes in Computer Science*. Springer, 2002.
9. M. Haj-Hussein L. Logrippo, M. Faci. An introduction to lotos: Learning by examples. *Comp. Networks and ISDN Systems*, 23(5):325–342, 1992.
10. K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.
11. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
12. S. N. Strubbe, A. A. Julius, and A. J. van der Schaft. Communicating Piecewise Deterministic Markov Processes. In *Proceedings IFAC Conference on Analysis and Design of Hybrid Systems ADHS 03*, 2003.
13. S. N. Strubbe and R. Langerak. A composition operator for complex control systems. Submitted to Formal Methods conference 2005, 2005.

14. S. N. Strubbe and A. J. van der Schaft. Stochastic equivalence of CPDP-automata and Piecewise Deterministic Markov Processes. Accepted for the IFAC world congress, 2005.
15. S.N. Strubbe and A.J. van der Schaft. Bisimulation for communicating piecewise deterministic markov processes (cpdps). In *HSCC 2005*, volume 3414 of *Lecture Notes in Computer Science*, pages 623–639. Springer, 2005.
16. A.J. van der Schaft. Bisimulation of dynamical systems. In *HSCC 2004*, volume 2993 of *Lecture Notes in Computer Science*, pages 555–569. Springer, 2004.